High availability - Solutions and implementations

White paper

High availability - Solutions and implementations

With ever-increasing pressure to remain online and attend to customer needs, IT systems need to constantly review and adapt their solutions on how to cope with system failures and resume operation in the shortest amount of time, with minimum to no material/data loss. This is reflected in the variety of High Availability solutions available and number of business case scenarios that they address.

This white paper provides an overview of the operation and implementation of various High Availability (HA) solutions that address individual business issues that organizations face. The implementation of available solutions is discussed using PostgreSQL's native replication technology, with third-party utilities when a limitation exists in the native solution.

Introduction

The readers of this document are advised to assess each implementation against their business requirements before proceeding to deploy any of them to a production environment.

The readers are also advised to direct any queries regarding this publication to their Fujitsu Sales representative so they can be addressed by one of our PostgreSQL consultants.

Background

The evolution of business requirements has resulted in changes and enhancements to Information Technology software and hardware. The general high availability requirements have included ease of management, safety against data loss, fault tolerance, instant recovery, robustness, security, data integrity, data redundancy, and minimal downtime during maintenance.

This white paper discusses how high availability, using PostgreSQL and other related third-party utilities, can be utilized to tackle some of the difficulties that organizations face in making databases always available to satisfy their requirements.

A complete high availability solution for databases should be able to target factors such as high resilience, data consistency, and instant availability following a failure for business continuity. Some of the major requirements of HA solutions also include load sharing and balancing, and fast failover versus controlled switchover, which dictate how a database system should be configured for performance, data protection and availability.

High Availability requirements

In order to function continually without breakdown, an online business requires high availability of its IT systems and databases.

In turn, to be continuously available for business continuity, a database should be fault-tolerant and robust to withstand failures. Fault tolerance and robustness are achieved by configuring a high availability system in one of the ways discussed in this white paper to satisfy business requirements.

Disaster recovery

Disaster recovery (DR) is the capability to bring an IT system back online following an outage. In terms of database technology, this translates to ensuring that a database is fully recovered following a failure.

A mission-critical database system should always be available to satisfy requests from applications at all times. This literally means "zero downtime", which is only achieved by maintaining a secondary copy of the database in a different location, ready to satisfy application requests in case the primary copy fails to come online in a given timeframe. The secondary copy is kept in standby mode, and offers an ideal disaster recovery configuration. Synchronization of the secondary copy with the primary database is kept with the help of replication.

An appropriately configured DR database ensures very tight and short Recovery Point Objective (RPO) and Recovery Time Objective (RTO) targets resulting in minimal data loss and quick recovery.

- RPO: Measure of minimizing the amount of data loss.
- RTO: Measure of minimizing the time taken to perform database recovery.

Load sharing and balancing

This is the distribution and balancing of processing over two or more systems to achieve optimal resource utilization.

Highly critical databases are often required to respond to a considerable number of READ queries and at the same time execute a substantial number of WRITE transactions. READs and WRITEs will be performed concurrently, and the database will experience an immense load to satisfy the application requests. This will result in resource contention, followed by performance degradation and server crashes.

A perfect solution to alleviate this problem is to separate READs from WRITEs. High-volume READs generally occur when large reports such as month-end, quarterly, half-yearly and annual reports are generated, all of which tend to be resource-consuming and take a long time to process.

Enterprise database systems demand high availability - systems should be tolerant of failures and operate stably at all times. General high availability requirements include ease of management, safety against data loss, fault tolerance, instant recovery, robustness, security, data integrity, data redundancy, and minimal downtime during maintenance.

On the other hand, WRITEs generally occur during normal processing of a business, such as when new customers register themselves in the system, records are updated, or discontinued products are deleted, all of which require a short time to process, and are instantaneous and concurrent.

Routing all READs to a real-time replicated copy of the database will ensure that resources are not locked, thereby allowing WRITEs to occur on demand on the primary database.

Sharing and balancing the load is also beneficial for horizontal scalability, when a business grows across multiple geographical regions thus requiring additional databases, either in the form of one-to-many or cascaded configuration. It ensures that operations are balanced based on load across synchronized databases.

Instant failover

This requirement refers to the capability to bring a replicated system online without delay following a crash of the primary system.

A database may suffer an outage at any time and crash due to power supply disruption, hardware failure, software component error, or simply as a result of human mistake. The amount of time that will be required to bring the database back online will depend on the severity of the failure.

If the cause of the server crash was due to power failure, then it can be rectified immediately by restarting the database, but even such a short-lived outage can cause a significant loss to the business. On the other hand, if the failure is caused by any of the other factors mentioned above, then the outage may be longer, ranging from hours to days.

A copy of the database, configured to immediately assume the role of primary database in such an event minimizes the business risk and downtime, satisfying the instant failover requirement of a database system.

Controlled switchover

This is the manual process of switching over an IT system to a replicated system.

IT software systems are upgraded and patched to latest versions to address any discovered security vulnerabilities, increase processing power for efficiency and enhance programing code, or to just include additional functionality. Some of these operations can take from a couple of minutes to a few hours, and a database server, being a software component, cannot afford to be unavailable for that long.

A real-time replicated database system configured for this purpose should be able to switch over between a primary system and standby one instantly without malfunction, and later switch back to the initial configuration.

This manual switchover and switchback of the database paves way for maintenance tasks such as rolling patches and upgrades. A faster switchover-backed highly available database system offers maximum serviceability and 100% uptime.

High availability modes

The High Availability requirements discussed above can be achieved using the replication modes explained below. The replication modes satisfy one or more requirements for business continuity and contingency planning.

Cold standby

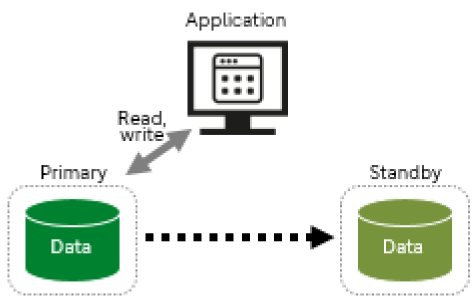
In this mode the standby database system will be in an inactive state, which will be started when the master system crashes following a major failure of hardware, software, power supply or the complete site. This involves copying nightly backups of the master database to a standby server and copy the transaction logs (WAL) until the next complete backup set is copied over.

In the event of a failure of the master database, the standby database will be initiated in recovery mode and all available transaction logs will be applied up to the current point in time, to recover it to be as close a copy of the master database as possible at the point of failure.

Organizations must ensure the availability of their systems in order not only to disrupt operations, but also to avoid losing revenue, damaging their reputations, and frustrating users, who may take to social media to voice their complaints, further amplifying the problem.

There are no particular advantages of this mode over the other ones. It was the only disaster recovery mode that was available in the past, and had a major drawback in that it would take a long time for the database to be brought online, due to the large amount of transaction logs that needed to be applied to make it a current copy of the master database from the previous night's backup.

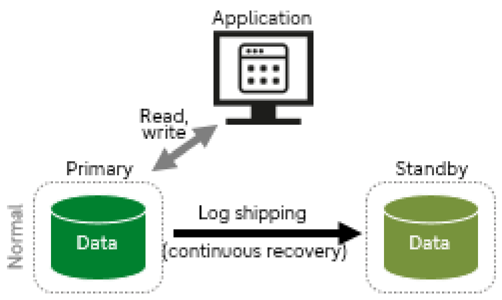
This technique is outdated and has been superseded by other implementations, listed below.



Cold standby

Warm standby

This is similar to the previous method, except that the standby database will be online, and the transaction logs will be continuously applied. This type of replication is also called "Log Shipping", as the transaction logs from the master are copied over to the slave and applied during normal operation.



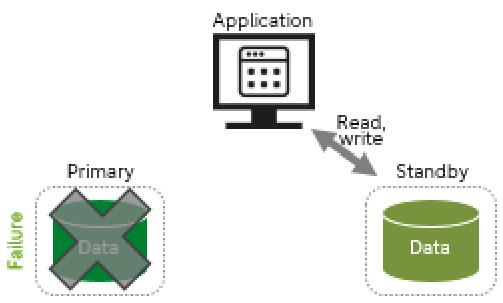
Warm standby (normal operation)

The advantage of this mode over cold standby is that the standby database is readily available to be failed over and made the new master database. However, the application will still need to be pointed to the new server, which might increase the RTO.

This limitation can be minimized with a floating IP address or a load balancer module which will become the single point of connection for the application. The load balancer will further decide whether to route incoming connections to the old master or the new one.

Although this mode satisfies the disaster recovery configuration of a HA system, there is still a major limitation of resource utilization due to the fact that in PostgreSQL implementation a slave server is not accessible even to read-only queries, which results in an idle server having the best of compute capacity not being operational.

This limitation is solved in the replication mode we will discuss next.



Warm standby (after failure)

Hot standby

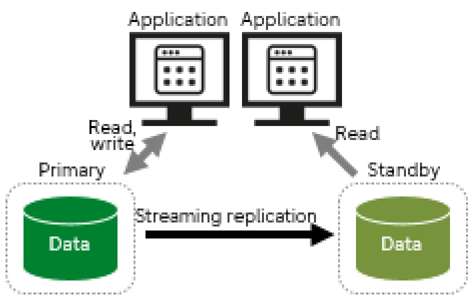
The primary objective of the hot standby mode is to address resource utilization, which was a limitation in warm standby. In this PostgreSQL setup, updated transactional records are continuously applied to the slave and it is also available for read-only queries. Such a standby with near real-time data and highest compute capacity can be utilized to generate complex reports during normal business hours, without much impact to the master database.

This effectively provides a setup of a master database that is busy executing OLTP load while the slave can be utilized for OLAP reporting.

In PostgreSQL, hot standby is achieved with the help of a "streaming replication" protocol, which allows the master to stream transactional records (WAL records) to the standby.

The primary objective of the hot standby mode is to address resource utilization. In this PostgreSQL setup, updated transactional records are continuously applied to the slave and it is also available for read-only queries.

This eliminates the need to wait for transactional logs to be completely filled with required data on the master, and then copied over to the slave server and replayed to the slave database.



Hot standby

This mode of HA satisfies both the disaster recovery and the load balancing and sharing configurations, but a customized failover mechanism can make it also satisfy the other two HA configurations: instant failover and controlled switchover.

Hot standby can run in either ASYNC or SYNC modes.

- In ASYNC mode, transactional data is streamed in a discontinuous (asynchronous) manner.

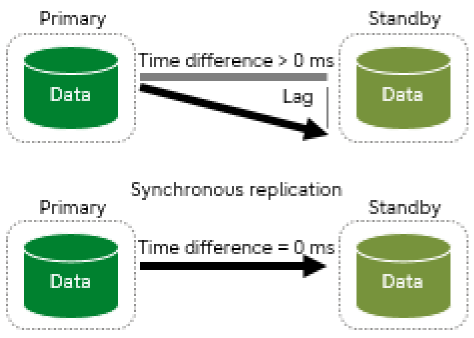
  Real-time replicated data can be achieved in such a setup, but it is not always guaranteed on very busy databases.

- In SYNC mode, transactional data is streamed synchronously from master to standby.

  Unlike ASYNC, a master database replicating data to a synchronous standby slave waits for acknowledgement that each transaction committed on the master has been replayed on the synchronous slave. This may cause the database system to have reduced serviceability.

  For instance, a two node master-standby setup using synchronous replication is likely to fail in case the network link between the master and the synchronous slave goes down - if this happens, the master is forced to wait for the unreachable slave.

  This will cause the master to wait indefinitely for transactional data to be applied and acknowledged by the slave before continuing to the next transaction.



Types of streaming replication

Streaming replication has a drawback, in that it may delete a transaction log from the master server without having copied and applied some of its records to the standby slave.

This can be addressed by Replication Slot. A replication slot ensures that no transaction log is deleted from the master while it may still be required by a standby slave, thereby diminishing the possibility of a standby slave becoming out of sync due to a missing log.

Logical replication

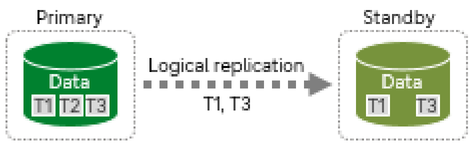
In this type of replication, the master sends individual transactions to the slave via a replication slot using the logical replication protocol.

Logical replication allows fine-grained control over both data replication and security, and uses a publisher-subscriber mechanism where the publisher is a master database and the subscriber is a standby slave receiving transaction commits.

Unlike the other replication modes, logical replication can be implemented between a master and standby slave of different major versions of PostgreSQL, starting from version 10 onwards. Additionally, logical replication is possible for just a database or a subset of a database, rather than for the entire database cluster, which is another limitation of the other replication modes.

Logical replication allows fine-grained control over both data replication and security, and uses a publisher-subscriber mechanism where the publisher is a master database and the subscriber is a standby slave receiving transaction commits.

It is possible to implement bi-directional replication using this mode, however the likelihood of query conflicts increases if the same data is configured to replicate in a bi-directional manner. Another potential issue is that there may be a delay in propagating committed changes to the destination.



Logical replication

This mode satisfies both the disaster recovery and the load sharing and balancing requirements, and allows the system to do without instant failover and controlled switchover to achieve high availability. This is because both the primary and standby databases respond to READs and WRITEs seamlessly when the data being accessed does not create conflicts.

If the application can switch between the primary and standby databases, because both the databases are open in READ-WRITE mode, then implementation of instant failover and controlled switchover can be ignored, and the application will eventually end up connecting to the server that is live and responding.

High Availability implementation

High Availability can be achieved in one of the following ways

- Active-passive (master-slave) database system
- Active-active (multimaster) database system

Let us discuss these modes and how to achieve them in PostgreSQL in conjunction with addressing some of the metrics already mentioned.

Active-passive database system

Since IT systems have evolved with the growth of business and the requirement of having disaster recovery to resume business following a failure, active-passive IT systems satisfied these demands and fulfilled some, if not all, metrics mentioned earlier.

A PostgreSQL database instance can be configured in one of the following setups.

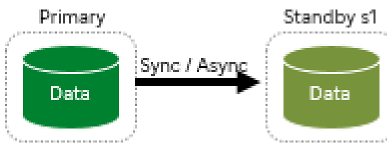
- Single master replicating data to a single slave
- Single master replicating data to multiple slaves
- Single master replicating data to a single slave which further cascades it to other slaves

Each active-passive setup implementation in PostgreSQL has one or more advantages over its immediate previous setup, and sometimes it even makes the previous setup obsolete. We discuss each setup and its characteristics below.

An application component can be configured in front of the databases so queries/transactions are routed to the appropriate databases based on the type of the query/transaction.

Single master-single slave

A single master database (also known as primary) replicates data to a single standby database (also known as slave). This setup can be used in cold standby, warm standby, hot standby, and logical replication.



Active-passive database system (single master, single slave)

As discussed under each implementation mode, it should be carefully assessed in order to ensure that it satisfies all business requirements, especially RTO and RPO. The best implementation in this setup is either an asynchronous hot standby or a logical replication.

A typical application type for this implementation would be a retail business management or hotel management, since the read-writes can occur on the master database, while the replicated database can be utilized for report generation. Although there is a buffer for the standby database to be unavailable for a short time or lag behind the master, occasional unavailability of the standby slave does not impact the serviceability of the master database.

The options for implementation of high availability can become complex depending on your organization's business requirements, since each alternative has its own advantages and trade-offs - each option must be carefully considered.

Failure of the master when the slave is unavailable can be disastrous, because the slave could be lagging behind the master from a few hours to more than a day, which may pose a real threat to business continuity.

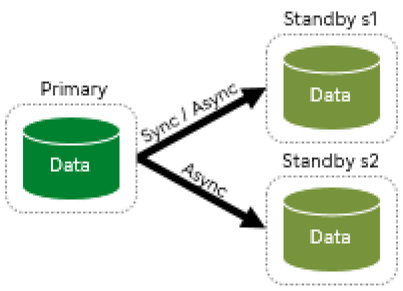
Single master-multiple slaves

A single master database replicating to multiple slaves is the best redundant system that applications can be configured with. This can be implemented with hot standby or logical replication.

This setup can be achieved in one of the following ways:

- One master replicating to two asynchronous slaves
- One master replicating to one synchronous slave and one asynchronous slave.

In both configurations, one of the slaves can run on the same data center as the master while the other runs in a different site, to eliminate the possibility of a single point of failure.



Active-passive database system (single master, multiple slaves)

Any application that is required to process a large volume of transactions, such as a banking system, stock exchange application or e-commerce portal is a use case of this type of setup.

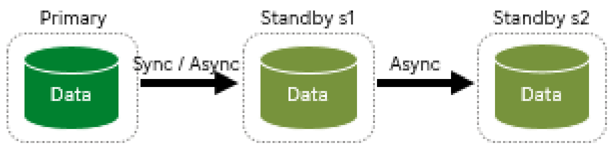
A master database will always be available for OLTP, while a real time replicating slave database could be a potential new master during an imminent failover. There is no limit to the number of slaves that can be added to a master, regardless of whether it is replicating synchronously or asynchronously.

This setup offers the maximum levels of database availability, performance, and data protection.

Single master-cascaded slaves

This type of setup is an extension of the previous implementation, in which a cascading slave is added either to the slave replicating asynchronously or to the one replicating synchronously.

The cascading slave relies on its upstream master (first level slave database) for transaction consistency. Additionally, the cascaded slave can only be configured to perform replication in asynchronous mode from its upstream master.



Active-passive database system
(single master, single slave to cascaded slave(s))

A simple way to remember which replication method can be set up for a slave is: if it is accepting data from the actual master, then it can be either asynchronous or synchronous. But if it is accepting data from another slave in a cascaded (upstream) manner, then the replication between these two slaves can only be set up as asynchronous.

As this is an extension of the previous setup, any application use case for the previous setup can be considered for this type of setup as well. There is a disadvantage in this setup regarding data lag - the cascaded replicating slave experiences a longer lag than its upstream standby slave, should the upstream standby slave lag behind the master.

Active-active database system

This is a multimaster setup that offers greater availability and extra resilience. It fulfils all metrics of a High Availability environment and checks all boxes when the business operation requirements are assessed against the IT system.

The active-active database setup allows writes from any node of the cluster. It makes it manageable for horizontal scalability to grow the cluster by adding nodes as business requirements evolve.

Business continuity, which includes Disaster Recovery and High Availability, can be defined as the ability to continue application operations even after outage, regardless or whether they are planned or not.

Active-active database systems should have an uninterrupted conflict detection and resolution mechanism in order to reflect the same version of data from any connected node.

This ensures high levels of data integrity during updates regardless of which node the updates originate from, so all participating nodes see the same data.

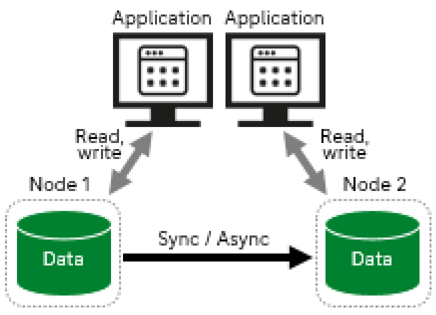
Asynchronous multimaster

Each server works independently, and periodically communicates with other servers of the cluster to identify conflicting transactions. Conflicts are either resolved by users or conflict resolution rules.

Synchronous multimaster

Each server is capable of handling write requests. Modified data is transferred from the server that handled the request to the other servers before the transaction is committed. This results in excessive table locks during heavy write load and may lead to poor write performance. Read performance is not affected, because read requests can be routed to and handled by any individual server.

To address this issue, some solutions use shared disk to minimize inter-node transfer overhead. This also eliminates the need to deploy workload partitioning or load balancer components.



Active-active database system (multimaster replication)

PostgreSQL does not offer this type of setup, but some tools make it achievable, as explained below.

An active-active database system can be implemented as follows.

Bucardo

Bucardo is an open source asynchronous replication system that implements a trigger-based solution with a Perl daemon that monitors requests and copies data back and forth between participating databases.

It manages a Bucardo database that acts as an inventory repository containing the replication rules, list of participating databases, details of connection to the databases, tables or database subsets being replicated, etc.

The replication rules are called "SYNCS", and are defined to copy the specific set of tables across the participating databases. The triggers defined on the replicating tables or subset of database store information about which rows were changed and should be sent to other participating databases.

There are limitations, such as neither DDLs nor very large objects are supported. Also, tables being replicated need a unique key defined on them, otherwise incremental replication of tables is not possible.

Bi-directional replication (BDR)

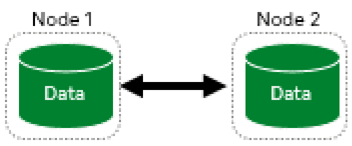
This is an asynchronous multimaster logical replication topology that can propagate committed writes from any participating database to all other participating databases. The writes are sent on a row-by-row basis. Each participating database in this mode is referred to as a *node*.

This topology can only be implemented from PostgreSQL 9.4 version onwards, and requires modification to the source code. Applications also require modifications to operate on a multimaster BDR setup, because they cannot be written as if they were connected to a single standalone server or single master system.

Data is not exactly the same on the participating nodes at any given point in time, because writes on a node are propagated to other nodes only after a commit on the original node.

The active-active database setup is a multimaster setup that offers greater availability and extra resilience. It allows writes from any node of the cluster, and makes it manageable for horizontal scalability to grow the cluster by adding nodes as business requirements evolve.

So, as long as the writes have not been committed on the original node, all other nodes will not see the most current version of data. This implementation utilizes the logical decoding feature to replicate across nodes.

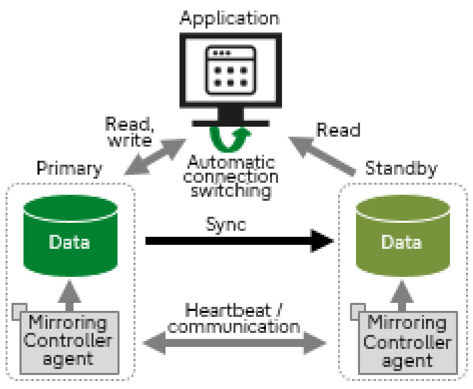


Active-active database system
(bi-directional replication)

Fujitsu Enterprise Postgres and High Availability

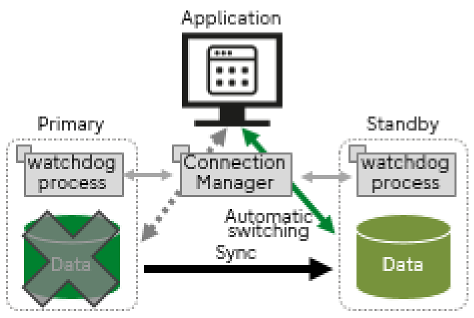
Fujitsu Enterprise Postgres is a PostgreSQL-based, highly reliable RDBMS, and is one of the fastest and most secure enterprise databases available. As a world-class mission-critical database, it ticks all the boxes of the HA configuration requirements.

- **Disaster recovery**: It is built on top of the native PostgreSQL streaming replication, which offers an excellent disaster recovery replica with real-time data synchronization.
- **Instant failover**: Managed by the flagship Mirroring Controller, it detects failures of server process, operating system, network, and disk to initiate failover from primary to standby.


Application
Read, write
Read
Primary
Standby
Automatic connection switching
Sync
Data
Data
Heartbeat / communication
Mirroring Controller agent
Mirroring Controller agent

Instant failover using Mirroring Controller

- **Controlled switchover**: Manual controlled switchover can be performed as needed, while switchback can be achieved with the help of an open-source tool such as pg_rewind.
- **Load sharing and balancing**: Real-time data synchronization allows load sharing and balancing for READs and WRITEs.
- **Connection Manager**: The process provides heartbeat monitoring and transparent connection features - applications are able to connect to the appropriate database server without having to be aware of the server state.



Transparent connection using Connection Manager

Conclusion

We hope that this white paper gave you a better understanding of the various possible implementations for high availability, alongside their benefits as well as their shortcomings.

Contact us

If you have any questions about how Fujitsu Enterprise Postgres ensures your system is up and running, or would like to discuss details of your data journey, feel free to contact us at enterprisepostgresql@fujitsu.com.

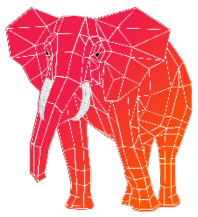
About Fujitsu

Fujitsu is the 5th largest IT service provider in the world, offering a full range of technology products, solutions, and services. Around 126,000 Fujitsu employees support customers in over 100 countries.

Fujitsu Enterprise Postgres can help your journey

Fujitsu Enterprise Postgres is the enhanced version of PostgreSQL, for enterprises seeking a more robust, secure, and fully supported edition for business-critical applications.

It is fully compatible with PostgreSQL and shares the same operation method, interface for application development, and inherent functionality. Designed to deliver the Quality of Service (QoS) that enterprises demand of their databases in the digital world, while supporting the openness and extensibility expected of open source platforms, all at a lower cost than traditional enterprise databases.



Fujitsu Enterprise Postgres

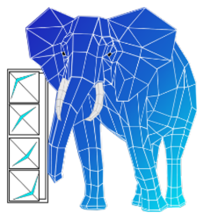
Combine the strengths of open-source PostgreSQL with the enterprise features developed by Fujitsu.

Enhanced speed, security, and support — without the costs associated with most proprietary systems.

Fujitsu Enterprise Postgres for Kubernetes

Utilize operator capabilities for provisioning and managing operations on the OpenShift Container Platform.

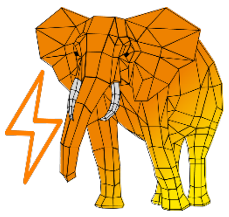
Business-ready database that integrates container operation technology for rapid development-to-production deployments.



Fujitsu Enterprise Postgres on IBM LinuxONE™

World-class platform that embraces open source and improves data security, performance, and business continuity.

The best of open source flexibility with the peace of mind that comes from knowing it is backed by Fujitsu and IBM.



Fujitsu Enterprise Postgres on IBM Power®

Experience frictionless hybrid cloud that can help you modernize to respond faster to business demands.

Fujitsu database designed for security, performance, and reliability, combined with IBM server built for agility in the hybrid cloud.

Discover how Fujitsu Enterprise Postgres' unique and enhanced features take PostgreSQL to the next level to provide enterprise-grade security, scalability, security, and performance.

Visit fast.fujitsu.com/key-features

Contact
Fujitsu Limited
Email: **enterprisepostgresql@fujitsu.com**
Website: fast.fujitsu.com

2022-09-08 WW EN