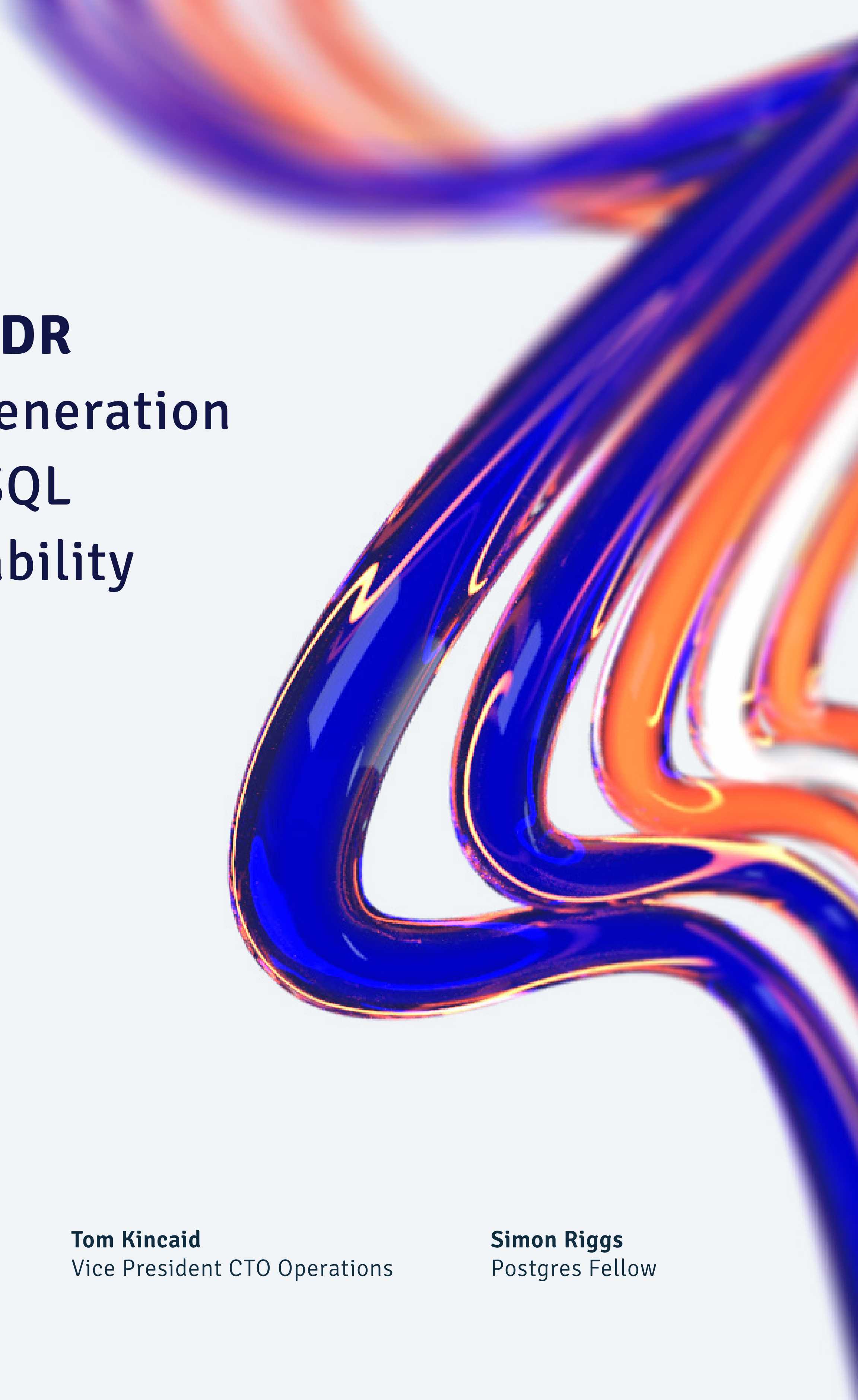
Postgres-BDR
The Next Generation of PostgreSQL High Availability



AUTHORED BY:

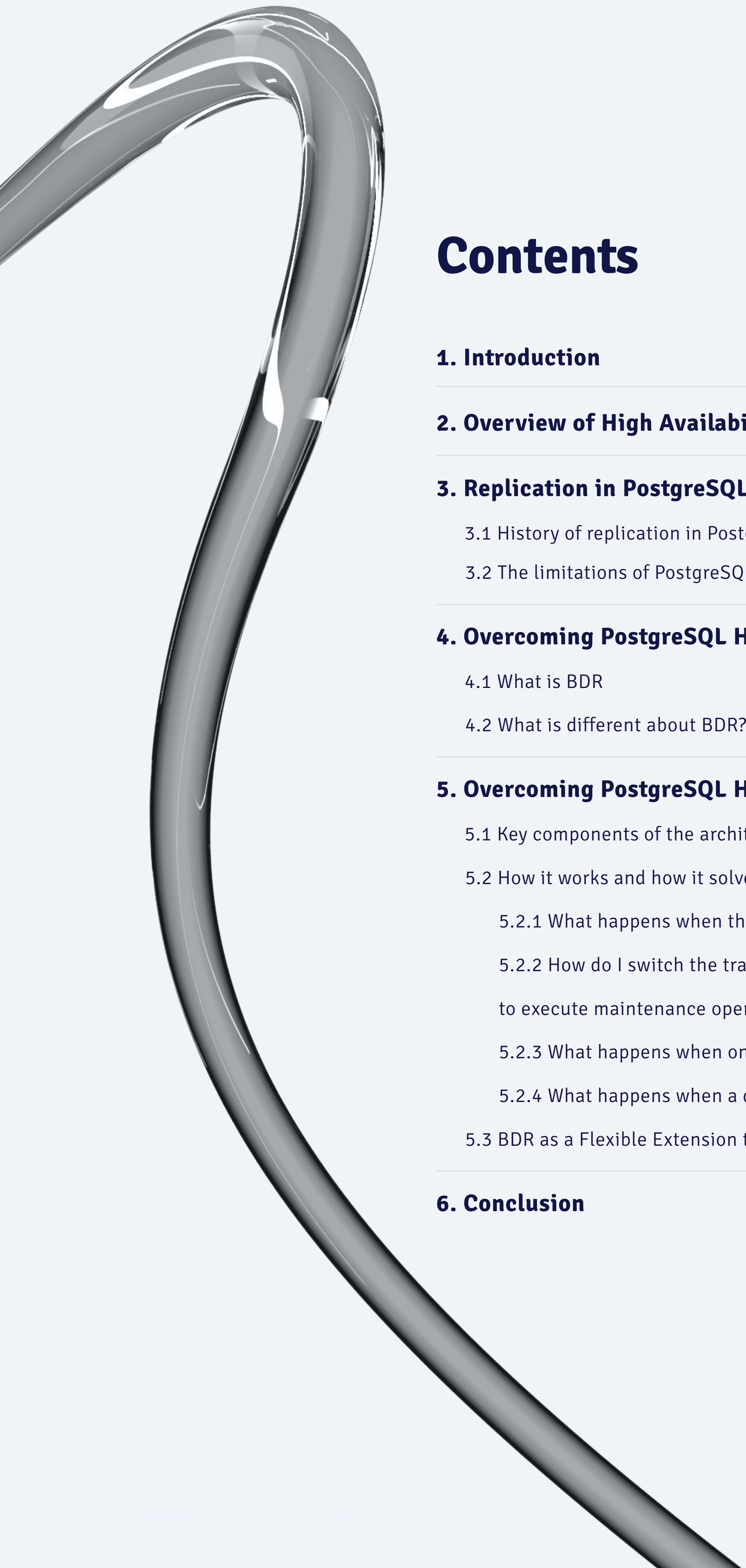
Marc Linster
Chief Technology Officer

Tom Kincaid
Vice President CTO Operations

Simon Riggs
Postgres Fellow

POWER TO POSTGRES

EDB™



Contents

1 Introduction

Over the last five years, the definition of High Availability (HA) has changed. HA used to refer to technology protecting users from hardware failures, network glitches, and software faults. Today, HA technology makes sure that software services are always on—365 days a year, 24 hours a day. HA products still protect users from failures, but as hardware, networks, power supplies, and storage devices have become much more reliable, near-zero downtime maintenance and management have moved to the forefront of the HA debate.

Near-zero downtime, or "Always On," has become a must-have for successful digital transformation in a global economy.

Postgres Bi-Directional Replication (Postgres-BDR™), often known as just BDR, has a wide field of possible applications, such as master data management, sharding, and data distribution. In this whitepaper, we only focus on high availability, where BDR provides unparalleled protection from infrastructure issues, combined with near zero-downtime maintenance and management capabilities. The Postgres-BDR Always On architecture for PostgreSQL has become the industry-leading solution for highly available PostgreSQL.

This white paper introduces the BDR technology and the Postgres-BDR Always On architecture.

2 Overview of High Availability in PostgreSQL

There are two fundamental approaches to high availability (HA) in ACID-compliant database systems: (1) shared storage with interconnected memory and (2) database replication. Oracle Real Application Cluster (RAC) is an example of the shared storage approach, which often requires expensive hardware to provide good performance. PostgreSQL uses a replication method, which allows PostgreSQL HA solutions to run on commodity hardware and in every cloud.

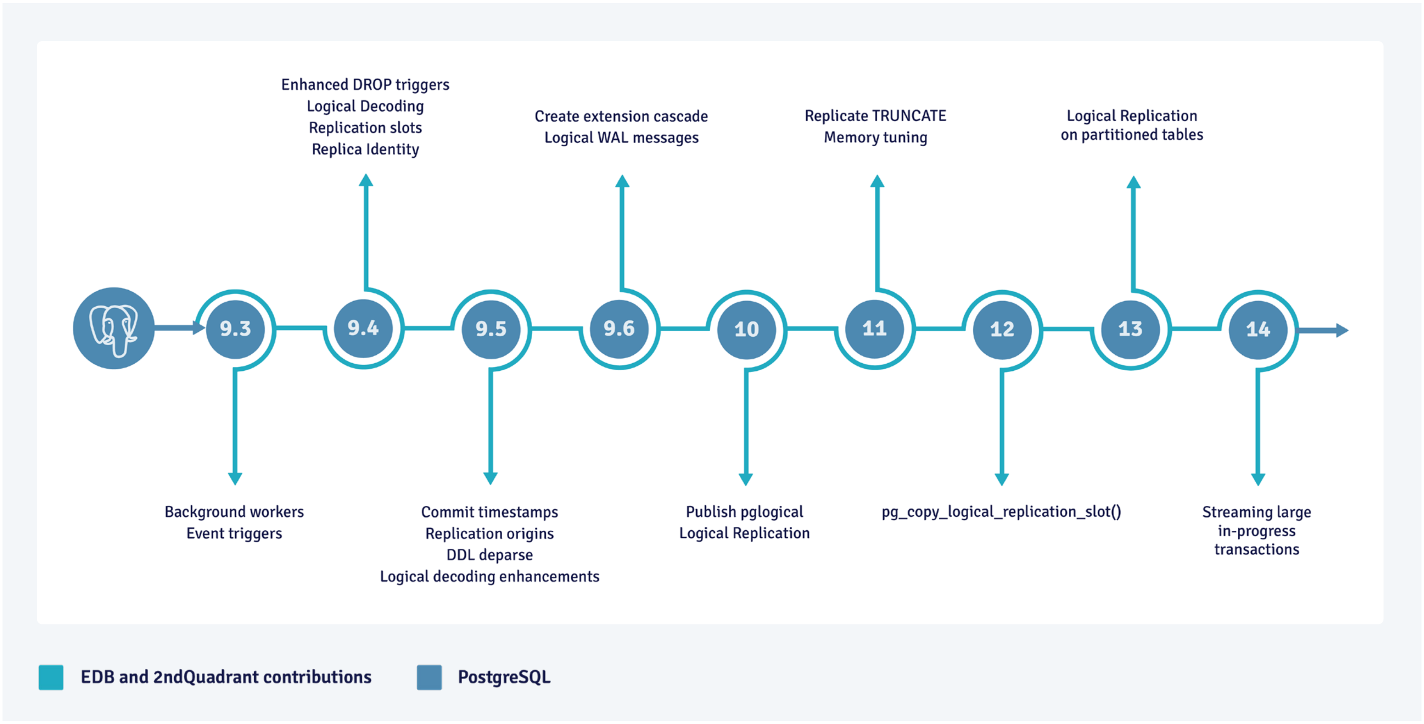
Traditional PostgreSQL HA architectures leverage physical streaming replication, a binary mechanism in which a primary server continuously sends database transaction logs, or write ahead logs (WAL), to a standby server. This ensures that the standby has a copy of the transactions the primary has executed. In case of a failure on the primary server, the standby could easily be promoted to become an active primary and resume the service. **EDB Failover Manager (EFM)** and **repmgr** are industry-leading technologies supporting up to 99.99% availability for PostgreSQL.

HA with streaming replication is limited in two key ways: (1) in case of failure, it takes a minimum of 20-30 seconds to identify and verify the failure, fence the old primary, and promote the standby to become the new primary; (2) the primary server and the replica need to be binary compatible, which means they must be on the same major PostgreSQL version, which in turn requires that both the primary and standby be taken offline together during an upgrade. Together, these limitations constrain streaming replication to applications that require 99.99% of availability and can have planned downtimes for major upgrades.

HA based on logical multi-master replication changes the equation: (1) each member of an HA cluster can accept transactions at any time, so that in the case of a failure of the current primary, the application can immediately start transacting on another server without first waiting for the cluster infrastructure to ascertain the failure and promote a replica; (2) logical replication supports PostgreSQL servers of different major versions, which means rolling upgrades are possible without ever shutting down the service. While logical replication has other advantages, these two capabilities allow us to use logical replication to create true Always On architectures that can achieve 99.999% availability.

3 Replication in PostgreSQL

3.1 History of replication in PostgreSQL



The very first replication and high availability solutions for PostgreSQL, Londiste and Slony, were external projects that used trigger-based logical replication.

PostgreSQL streaming replication was introduced in PostgreSQL 9.0 to improve the performance and reduce the latency of transaction log-shipping, which had been introduced in 8.2. Streaming replication also reduced the extent of data loss in case of a failure of the primary database server. PostgreSQL 9.1 introduced hot standby feedback and synchronous replication.

Subsequently, this infrastructure was built out to add the foundational capabilities for robust logical replication. For example: Event Triggers that fired on DDL statements, replication slots to coordinate activity on streaming standbys, replication of truncate statements, support of Logical Replication on Partitioned Tables, and so on. The EDB and 2ndQuadrant teams contributed and/or committed most of these features from the BDR project for the PostgreSQL community.

3.2 The Limitations of PostgreSQL Logical Replication for HA

PostgreSQL Native Logical Replication (PNLR) was introduced in PostgreSQL 10 in 2017. PNLR made major version upgrades possible without trigger-based technologies such as Slony or Londiste, which was a significant improvement over existing replication technology.

In 2021, there are still fundamental limitations with PNLR when being considered as part of highly available solutions. Examples of such limitations include but are not limited to:

- **DDL operations are not replicated. Operations such as creating and dropping tables need to be performed independently on each node.** Since there is no way to ensure transaction consistency of performing DDL operations, additional maintenance windows, where the database is receiving no database traffic, are required. This detracts from the system's ability to be highly available.

- **In PNLR there is no ability to fail over. If either the source or the target node goes down, during a heavy period of replication traffic, a significant risk exists that the entire replication process for a given set of tables will need to be restarted, and previously replicated data will need to be resent.** Therefore, PostgreSQL using PNLR are at greater risk than PostgreSQL physical replicas for having significant data loss when taking over as the primary data source for an application as a result of the current primary data source failing.

- **To properly identify updated and deleted rows during replication, logical replication systems require that each row in a replicated table have a primary key.** The primary key needs to be unique across all nodes in the cluster. PNLR provides no built-in mechanism to ensure that newly created and unique records on different nodes don't have the same primary key. The risk is that completely different records in a cluster are represented as the same record. Creating the assurance of unique primary keys is entirely the responsibility of the application developer. Therefore, for advanced cases such as High Availability, PNLR is error prone, which can result in data divergences and inconsistencies across the cluster.

- **PNLR is not integrated with backup and recovery solutions.**

- **While PNLR is a nice feature, it does not come with best practices and proven architectures for achieving common tasks, such as procedures to use PLNR for upgrades and maintenance operations.** Such operations need to be built and extensively tested for each deployment.

- **PNLR only replicates in one direction.** Therefore, when used in an upgrade scenario, if things go wrong after the upgrade and you encounter an issue, going back to a previous version of the database software is very complex and potentially impossible if not properly planned for in advance.

4 Overcoming PostgreSQL HA limitations with BDR

4.1 What is BDR

Postgres-BDR™ from EDB is a PostgreSQL database extension that enables extremely high availability for PostgreSQL clusters. It does this as a standard PostgreSQL extension through logical replication of data and schema in a mesh-based multi-master architecture, plus a robust set of features and tooling to manage conflicts and monitor performance. This means applications with the most stringent demands can be run with confidence on PostgreSQL.

BDR is asynchronous by default, uses a mesh topology, and creates an active-active architecture based on PostgreSQL logical replication.

BDR was built by the team that contributed many of the foundational replication capabilities in PostgreSQL. It was designed in conjunction with customers to allow them to replace complex, costly legacy solutions with modern, highly available databases capable of rolling upgrades.

4.2 What is different about BDR?

BDR takes PostgreSQL replication to the next level. It builds on logical replication and creates a robust multi-master capability that includes:

- Automatic DDL and DML replication in an active-active mesh network
- Failover and switchover infrastructure to re-route traffic in case of failures or during maintenance operation
- Advanced conflict detection and conflict management
- Management of distributed sequences
- Differentiated replication sets to control which data gets replicated and to which downstream databases

- Cluster expansion/consolidation
- Rolling database software upgrades
- Rolling schema change/migration using cross-schema replication
- Recovery from user error through solid integration with backup and recovery tools
- Improved security model making sure that all changes get replayed with minimal security privileges

BDR has a wide field of possible applications, such as master data management, sharding, and data distribution. **In this whitepaper, we only focus on BDR for high availability, where BDR delivers several additional features that allow administrators to control the latency/consistency trade-off:**

Column-Level Conflict Resolution. Ability to have per column level granularity for conflict resolution so that UPDATEs on different fields can be 'merged' without losing either change.

Conflict-free Replicated Data Types. Mathematically proven data types with demonstrated consistency in asynchronous multi-master update scenarios, for when conflicts are expected.

Transform Triggers. Filters, modifies or transforms incoming data with triggers activated on data before they are applied.

Conflict Triggers. Custom conflict resolution techniques that can be implemented with triggers that are activated when a conflict is detected.

Eager Replication. Provides conflict free replication by ensuring that all nodes can commit before allowing the local node to commit.

Commit At Most Once (CAMO). A consistency feature that helps an application understand whether a commit was received or the transaction aborted, even across single node failure. That way an application can be written to be idempotent and need never miss a transaction or commit it more than once, while remaining available most of the time.

Timestamp-based Snapshots. Provides consistent reads across multiple nodes for retrieving data as they appeared or will appear at a given time.

Rapid node join (bdr_init_physical). Utility to rapidly join nodes into the cluster using physical replication; important for large PostgreSQL databases.

LiveCompare. High speed verification utility to confirm that replication is exactly accurate across nodes, working with multiple PostgreSQL databases, across BDR nodes and also against Oracle. Designed and integrated with BDR for continuous live usage.

5 Postgres-BDR ™ Always On

Postgres-BDR™ Always On is a highly standardized, proven, and reliable PostgreSQL architecture for achieving the 99.999% HA goal of database availability on premises or in the public cloud. Always On is available in several variations. In this paper we focus on the variation that is designed to provide local HA in two redundant data centers—an active data center and a passive data center—both configured identically. Other variations of the Always On architecture include, but are not limited to: Single Data Center, Single pgBouncer/HAProxy per data center, and 'No Logical Standbys'.

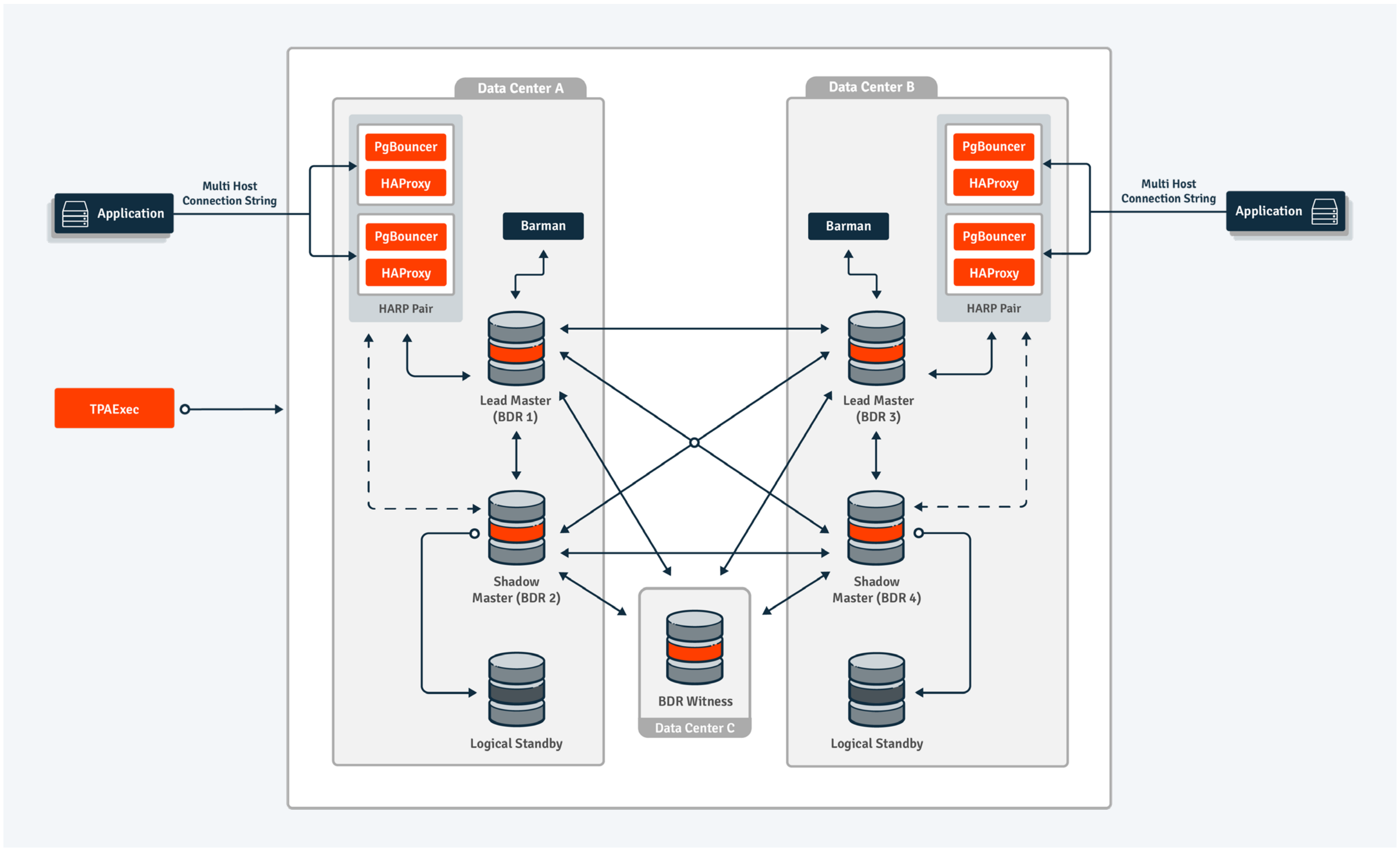


*Figure 1: **Postgres-BDR™ Always On** architecture for local high availability in two data centers*

5.1 Key components of the Postgres-BDR™ Always On architecture

Lead Master. A PostgreSQL server in read/write mode with the BDR extension installed. The Lead Master receives the write transactions and the read requests from the application. It is in a logical replication relationship with all other BDR nodes.

Shadow Master. The Shadow Master is a read/write enabled PostgreSQL server that does not currently receive traffic from the application, but it continuously replicates from the Lead Master and all other BDR nodes. In case of failure or switchover, the Shadow Master can take over almost immediately for the Lead Master.

Logical Standby. The Logical Standby is a read-only PostgreSQL node that replicates from the Shadow Master. It can be used for offloading of read transactions. Its primary role is to replace one of the masters in case of hardware failure, so that local HA can be re-established quickly.

HARP. The High Availability Router for PostgreSQL (HARP) makes sure that in case of a failure of the Lead Master, all subsequent database connections are successfully and reliably redirected to the Shadow Master.

pgBouncer. pgBouncer provides connection pooling to allow multiple application instances to connect to the BDR databases. Depending on the usage pattern, it can be configured in session mode or in transaction mode.

HAProxy. HAProxy provides connection routing and is tightly integrated with HARP.

Multi-host connection string. Applications use a multi-host connection string to allow for rapid failover to a second pgBouncer/HAProxy pair in case of a hardware failure.

Barman. The Barman server provides backup and recovery, especially for PITR use cases to protect against operator error or data corruption.

BDR Witness Node. A witness node is positioned in a third location to provide a quorum in case of a data center failure.

TPAExec. EDB's tool to deploy and operationally manage trusted PostgreSQL architectures, such as Always On.

EDB PostgreSQL Enterprise Manager® (PEM [not depicted in Figure 1]) can monitor BDR and its components Barman, pgBouncer, and HAProxy.

5.2 How it works and how it solves problems

Postgres-BDR™ Always On handles failures, switchovers, and other maintenance operations within the 99.999% HA boundaries. Here we illustrate a few of the most common scenarios.

5.2.1 What happens when the Lead Master fails?

HARP detects the failure of the node and switches all traffic at the pgBouncer/HAProxy level to the Shadow Master. Failure detection and switching of the traffic happens within a few seconds. As the Shadow Master is able to receive update transactions at any time, service resumes almost immediately.

5.2.2 How do I switch the traffic from Lead Master to Shadow Master to execute maintenance operations?

Using TPAExec, the administrator drains all connections on pgBouncer, and then reconnects pgBouncer to the Shadow Master. The use of TPAExec helps to ensure that all Postgres-BDR™ Always On components are aligned during this operation.

5.2.3 What happens when one of my pgBouncer/HAProxy nodes fails?

The applications use a multi-host connection string. When the server at the primary address in the connection string fails, then the application can immediately connect to the second pgBouncer/HAProxy server. HARP makes sure that both pgBouncer/HAProxy pairs point to the same BDR master and traffic immediately resumes.

5.2.4 What happens when a data center fails?

The management plane of the application will either redirect the application connections to the secondary data center, or—what is generally considered best practice—all inbound traffic is redirected to application servers in the second data center.

5.3 BDR as a Flexible Extension to PostgreSQL

BDR is implemented as a PostgreSQL extension and is available in two versions: Standard Edition (SE) and Enterprise Edition (EE). BDR 3.7.9 and later supports **EDB PostgreSQL Advanced** in Oracle compatibility mode or in standard mode, **EDB PostgreSQL Extended** (the former 2ndQuadrant PostgreSQL), and **open source PostgreSQL.**

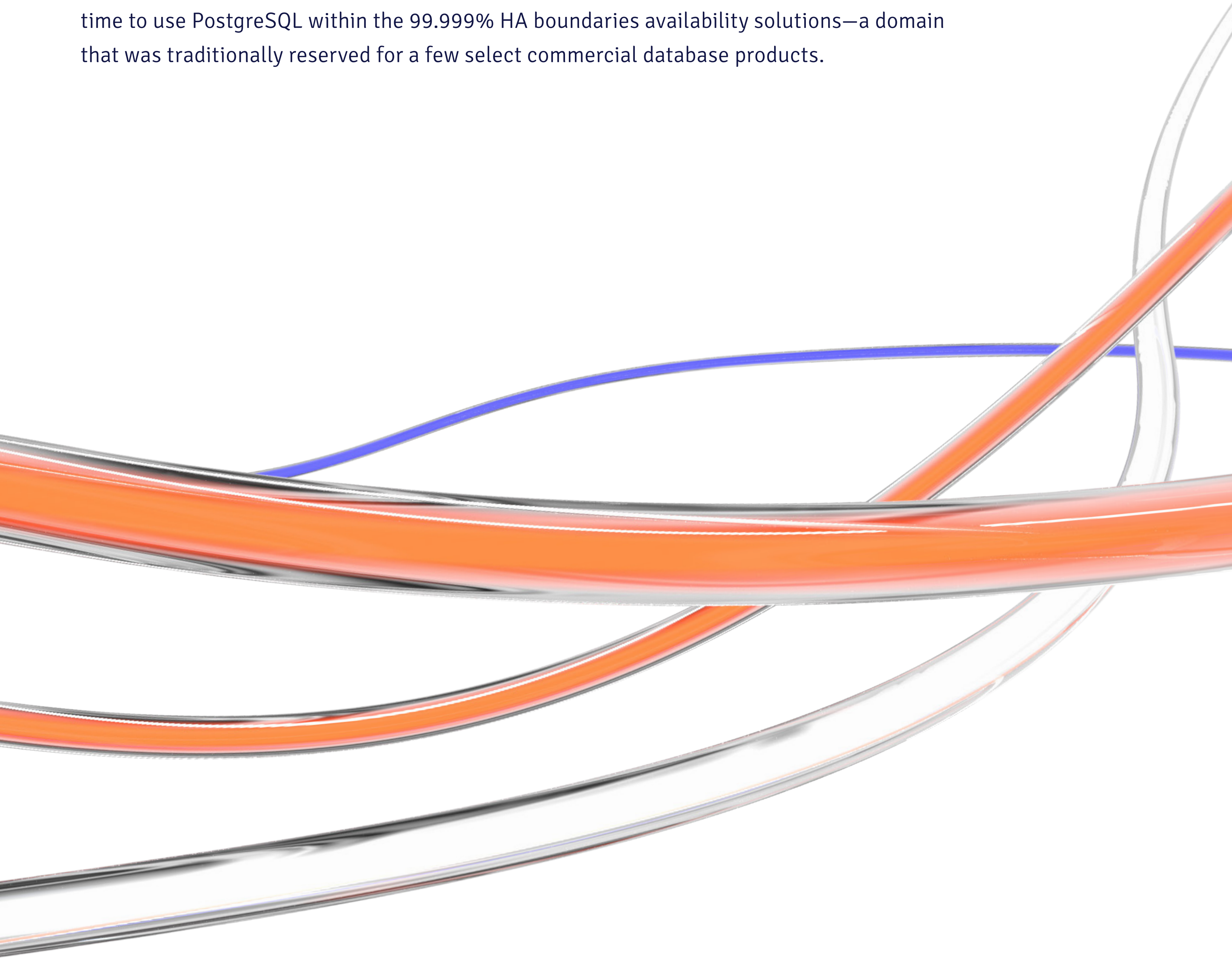
While Standard Edition meets many needs, Enterprise Edition provides additional

- **Performance.** BDR-EE provides Parallel Apply to greatly enhance replication throughput.
- **Monitoring.** BDR-EE enhances monitoring with timing estimates for replication lag.
- **Robustness.** BDR-EE provides Commit At Most Once functionality that ensures idempotence of transaction changes, a technique that is important for example when handling high-value financial transactions.
- **Production Control.** BDR-EE has an extensive range of additional features that are important for distributed and/or high concurrency applications, such as CRDTs, advanced conflict handling options, and advanced DDL handling options for production databases.

Database Distribution	BDR Standard Edition	BDR Enterprise Edition
PostgreSQL 11, 12, 13	✓	
EDB Postgres Extended 11, 12, 13		✓
EDB Postgres Advanced 11, 12, 13 with Oracle compatibility	✓	✓

6 Conclusion

EDB Postgres-BDR™, with the Always On architecture, is the industry leading solution for PostgreSQL High Availability. Postgres-BDR™ Always On allows customers for the first time to use PostgreSQL within the 99.999% HA boundaries availability solutions—a domain that was traditionally reserved for a few select commercial database products.



About EDB

PostgreSQL is increasingly the database of choice for organizations looking to boost innovation and accelerate business. EDB's enterprise-class software extends PostgreSQL, helping our customers get the most out of it both on premises and in the cloud. And our 24/7/365 global support, professional services, and training help our customers control risk, manage costs, and scale efficiently.

With 16 offices worldwide, EDB serves over 4,000 customers, including leading financial services, government, media and communications, and information technology organizations. To learn about PostgreSQL for people, teams, and enterprises, visit EDBpostgres.com.

Postgres-BDR The Next Generation of PostgreSQL High Availability

EDB™