EDB + Red Hat

High Availability & Disaster Recovery of PostgreSQL Databases with **EDB and Red Hat OpenShift**

Reference architecture for a single Red Hat OpenShift cluster

enterprisedb.com

Although EDB Postgres Distributed for Kubernetes (PGD4K) supports architectures spanning multiple Red Hat OpenShift clusters, this document focuses only on elevating the single point of failure of PostgreSQL databases with EDB PG4K to a Red Hat OpenShift cluster, typically equivalent to an entire region in the cloud or a data center in on-premises deployments. The recommendations included in this document serve as a building block for more complex architectures across different Red Hat OpenShift clusters. A separate document covers addressing multiple Red Hat OpenShift clusters.

Running business-critical PostgreSQL databases on Red Hat OpenShift

EDB Postgres for Kubernetes, or EDB PG4K, is a certified Level 5 Operator for Red Hat OpenShift, designed to streamline Day 2 operations of PostgreSQL databases. It enhances database management with features such as high availability, disaster recovery, primary/standby cluster management, automated failover, self-healing, and online continuous backups to object stores or volume snapshots. Additionally, it supports point-in-time recovery (PITR), ensuring robust data protection and recovery options, seamlessly integrating with business continuity solutions such as Red Hat OpenShift API for Data Protection (OADP) and Veeam Kasten, Trilio, Portworx Backup, IBM Fusion, and others.

EnterpriseDB (EDB) has long been a leader in PostgreSQL development. Now, as founding sponsors and maintainers of the open source CloudNativePG operator—which sits at the heart of PG4K—EDB is pushing the boundaries of innovation in cloud-native database environments.

With the combined power of Red Hat OpenShift and EDB PG4K, you can run your PostgreSQL business-critical databases with prime RPO and RTO goals in a cloud-native environment, either as microservice databases next to their applications or as a database-as-a-service (DBaaS) solution serving internal organizational needs or public customers. Red Hat OpenShift and EDB PG4K create unprecedented opportunities for cloud deployments, whether private, public, hybrid, or multi-cloud. The stack offers minimal differences in configuration and management across various environments, providing flexibility, enabling data portability, and eliminating vendor lock-in at the cloud service provider level.

Understanding your current Red Hat OpenShift architecture

Ensuring business continuity requires a thorough understanding of your infrastructure's single points of failure (SPoFs), their locations, mitigation methods, and potential impacts on downtime, productivity, and reputation.

Red Hat OpenShift, and Kubernetes more broadly, is designed to provide high availability and self-healing capabilities within a single cluster. It can operate, where available, across multiple data centers or availability zones (AZs) and on various nodes (virtual or physical machines), offering resilience against component failures. Figures 1 and 2 illustrate typical Red Hat OpenShift deployments in cloud environments and on-premises infrastructures, respectively.

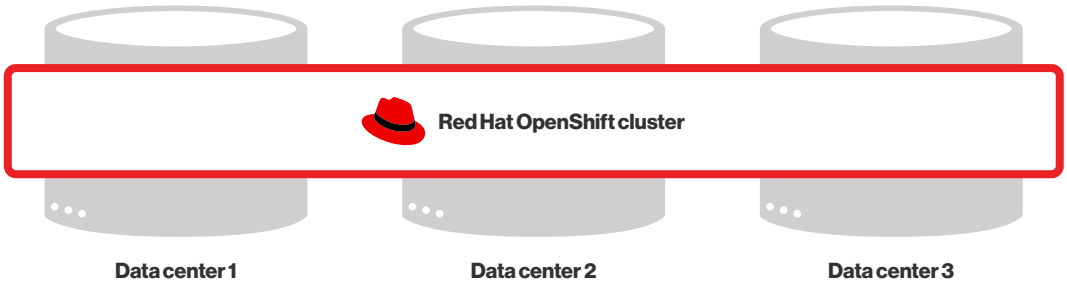

Figure 1. A Red Hat OpenShift cluster deployed in a "stretched" configuration across three cloud availability zones

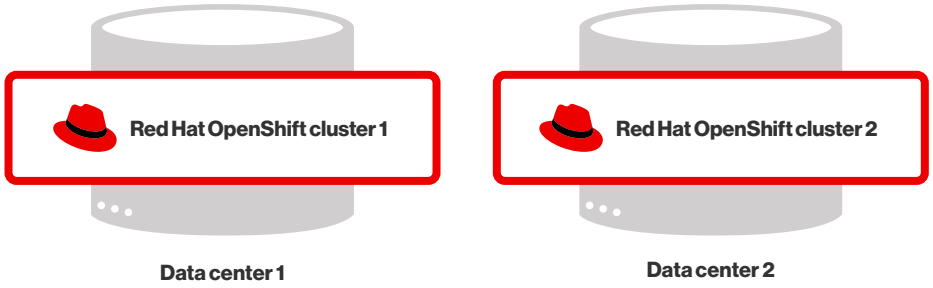

Figure 2. A standard Red Hat OpenShift on-premises deployment with distinct clusters for each data center

A standard Red Hat OpenShift deployment consists of three key types of nodes:

- **Master nodes (control plane):** Manage the cluster, handle scheduling, and run essential services such as the API server, scheduler, and controller manager.
- **Infrastructure nodes:** Host critical infrastructure components such as the default router, integrated container image registry, and cluster metrics and monitoring tools.
- **Worker nodes:** Run the application workloads, hosting the containers and executing application pods.

This architecture ensures efficient management and scalability for both cloud and on-premises deployments.

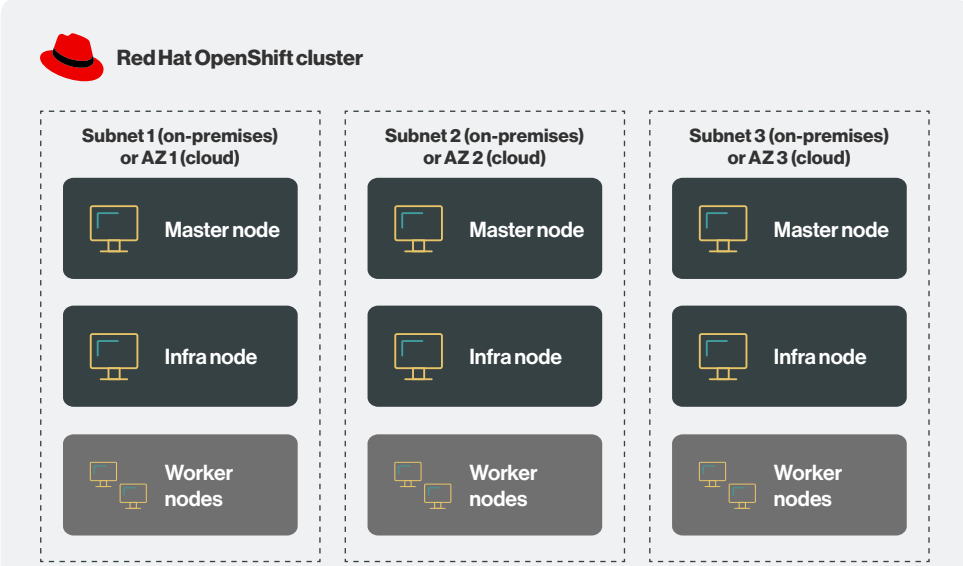

Figure 3. The three main kinds of nodes in a standard Red Hat OpenShift deployment

Architectural considerations for PostgreSQL deployment on Red Hat OpenShift

EDB PG4K extends the Red Hat OpenShift API to seamlessly manage a PostgreSQL cluster within the same Red Hat OpenShift cluster. It also provides ways to mitigate the business continuity risks across different Red Hat OpenShift clusters through native physical replication.

Key recommendations for managing PostgreSQL databases on Red Hat OpenShift

1. **Rely on multiple availability zones (cloud) or subnets (on-premises):** Leverage three availability zones/subnets within a Red Hat OpenShift cluster. This configuration ensures zero data loss for high availability and achieves a very low recovery time objective (RTO) within a single Red Hat OpenShift cluster.

2. **Isolate PostgreSQL workloads:** Dedicate nodes specifically for PostgreSQL workloads, separating them from other applications. Use Red Hat OpenShift features such as node labels, selectors, taints, and tolerations to enforce this separation through declarative configuration. Reserve at least three nodes for PostgreSQL in each Red Hat OpenShift cluster, distributed evenly across the availability zones/subnets. Scale in multiples of three to maintain balance and resilience. In some cases, you might dedicate three Red Hat OpenShift worker nodes to a single PostgreSQL cluster, each hosting a PostgreSQL instance (primary or replica).
3. **Trust PostgreSQL replication over storage replication:** Unlike many cloud-native applications that synchronize state at the storage level, PostgreSQL handles state synchronization independently through its built-in physical replication capabilities, based on write-ahead log (WAL) shipping. These capabilities, used successfully in production by millions of users worldwide for more than a decade, include asynchronous and synchronous streaming replication over the network, as well as asynchronous file-based log shipping, typically used as a fallback option (e.g., storing WAL files in an object store). Standby servers, or replicas, can also handle read-only workloads thanks to the hot standby feature.

The result is a complete and transparent physical separation of PostgreSQL workloads from other applications, with dedicated nodes for PostgreSQL, termed "Postgres nodes." These nodes can be easily added to existing Red Hat OpenShift clusters and scaled up as needed. This setup is illustrated in the following figure:

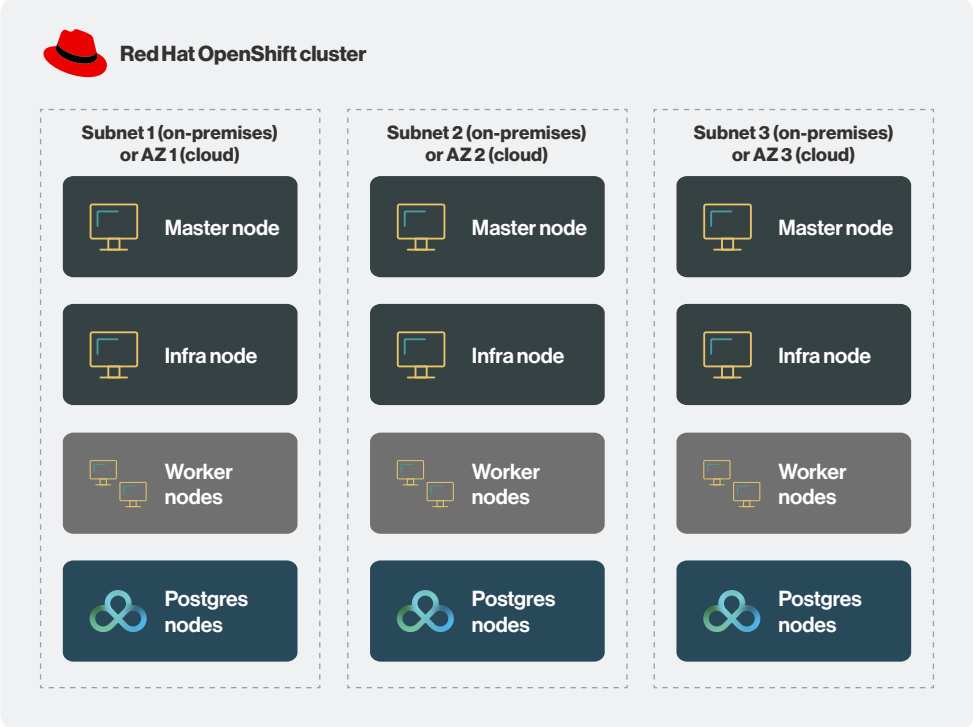

Figure 4. The four primary types of nodes in a typical EDB PG4K deployment on Red Hat OpenShift

To designate specific worker nodes as PostgreSQL nodes within your OpenShift cluster, we recommend using the `node-role.kubernetes.io/postgres` label. This label helps identify the nodes that should handle PostgreSQL workloads. You can apply the label to your chosen nodes using the following command:

```
$ oc label node <node-name> node-role.kubernetes.io/postgres=" "
```

You can set appropriate taints on these nodes to ensure that only PostgreSQL workloads are scheduled on your designated Postgres nodes. This will prevent other types of workloads from being scheduled on them unless they have the matching tolerations. For example, you can add the following taint to a PostgreSQL node:

```
$ oc adm taint nodes <node-name> node-role.kubernetes.io/postgres=:NoSchedule
```

By applying this taint, the node will only accept pods with a corresponding toleration, limiting it to PostgreSQL workloads. When creating them, remember to set the proper tolerations on your PostgreSQL clusters so they can be scheduled on the tainted nodes. This approach enhances resource isolation and ensures that your PostgreSQL workloads run in the most suitable environment.

Storage considerations

EDB PG4K relies on storage classes and directly manages file system PersistentVolumeClaim (PVC) resources instead of StatefulSets for finer control over database files.

Despite being storage agnostic, database workloads demand performance and data durability, which can only be ensured through proper benchmarking before production. In most cases, local storage attached to a worker node is preferable. Detailed storage recommendations are covered in a separate document.

Recommended PostgreSQL architecture

EDB PG4K leverages PostgreSQL native physical replication to:

- Manage the single primary/multiple standby PostgreSQL cluster within a single Red Hat OpenShift cluster for HA purposes using streaming replication, including synchronous replication; this setup provides automated failover, self-healing, and rolling upgrade capabilities.
- Implement continuous backup to an immutable object store, essential for full recovery and PITR with a maximum RPO of five minutes.

Moreover, EDB PG4K enables distributed active/passive topologies across multiple Red Hat OpenShift clusters and regions via object stores (RPO ≤ 5 minutes) and/or streaming replication (near-zero RPO, depending on latency) through the replica cluster feature.

Within a Red Hat OpenShift cluster, EDB PG4K provides a custom resource definition (CRD) called `Cluster` to manage a highly available PostgreSQL cluster. This cluster features a single primary and multiple replicas that are ready to become primary in the event of an unexpected incident (e.g., automated failover) or during planned operations (e.g., switchover following a Red Hat OpenShift node update).

For application routing, EDB PG4K automatically maintains a set of Red Hat OpenShift service objects, typically of `ClusterIP` type, that direct traffic to the primary or read-only replicas. These services can be easily configured through service templates, including load balancers to provide external access to the database from outside Red Hat OpenShift if needed.

EDB PG4K supports advanced scheduling capabilities, such as pod affinity and anti-affinity, node selectors, and tolerations. These features help manage single points of failure by ensuring that each PostgreSQL instance runs on different nodes and, where possible, in different availability zones/subnets.

A typical architecture of a single PostgreSQL cluster in high availability via quorum-based synchronous replication is depicted in the following diagram (availability zones/subnets are marked with dotted lines, as they might not exist):

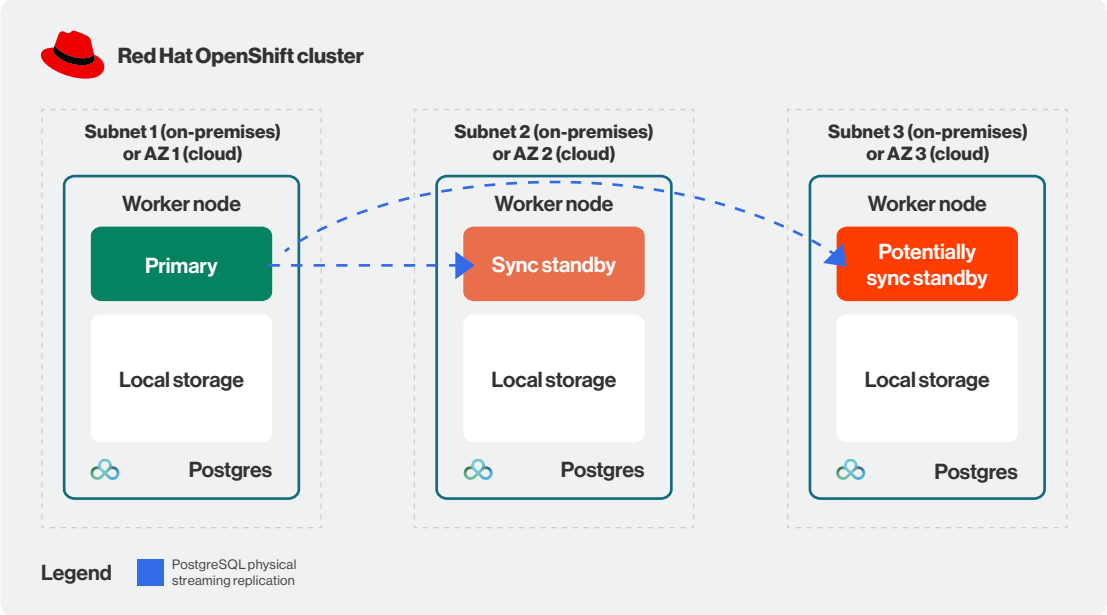

Figure 5. Typical architecture of a high-availability PostgreSQL cluster

By design, in the above case, every committed transaction in the database is guaranteed to be written to a replica using PostgreSQL's native quorum-based synchronous replication, ensuring an RPO of zero for high availability. Given that Red Hat OpenShift can immediately detect a failure on a primary, the failover time is typically just the time it takes for the most advanced replica to be promoted to primary status, usually within a minute in total. This makes achieving 99.99% uptime a realistic goal, given a solid underlying infrastructure.

Regarding disaster recovery, EDB PG4K allows you to set up continuous backups of a PostgreSQL cluster through base backups and the WAL archive, ensuring, by default, a maximum RPO of five minutes.

In the simplest scenario, both components can reside in a local object store, either using the native provider solution if you are in the cloud (e.g., AWS S3, Azure Blob Storage, Google Cloud Storage) or a specialized product such as Red Hat OpenShift Data Foundation (ODF) object storage. If your storage class supports them, EDB PG4K also allows you to perform backup and recovery using volume snapshots. Hybrid strategies with both object stores and volume snapshots having different retention policies are also possible.

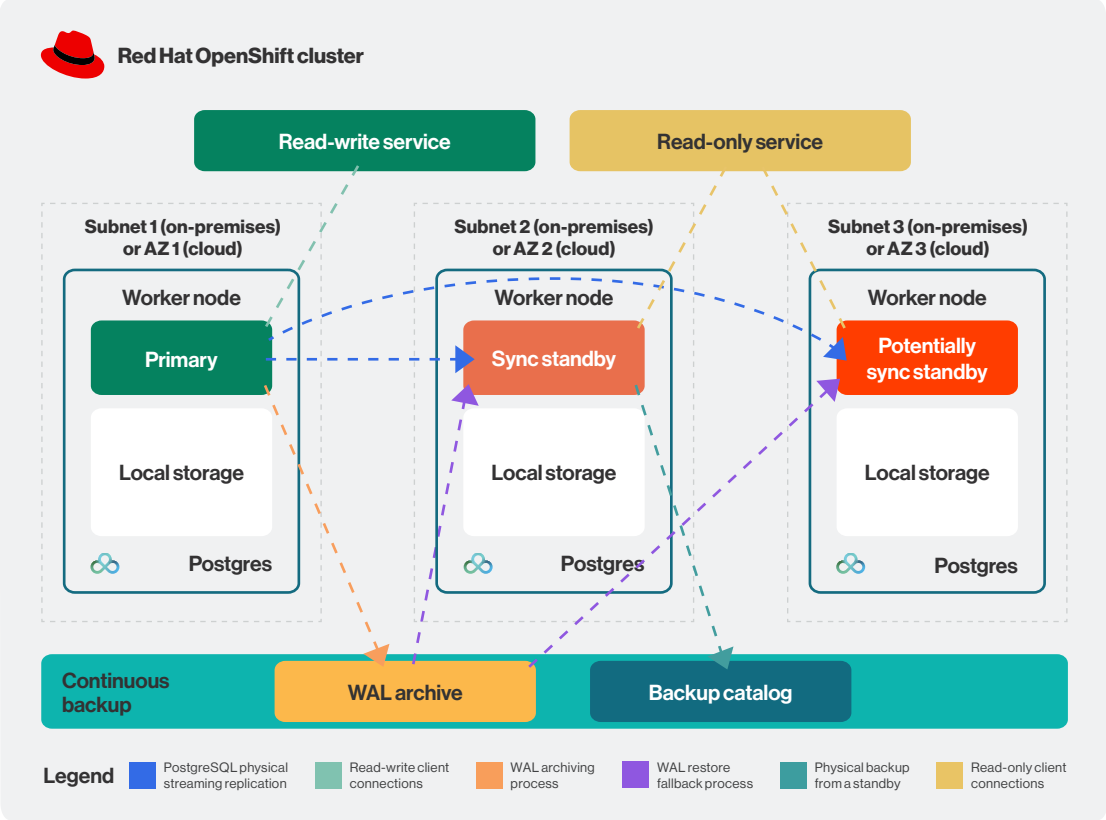

Figure 6. Typical architecture of a high-availability PostgreSQL cluster, highlighting services and continuous backup

Summary

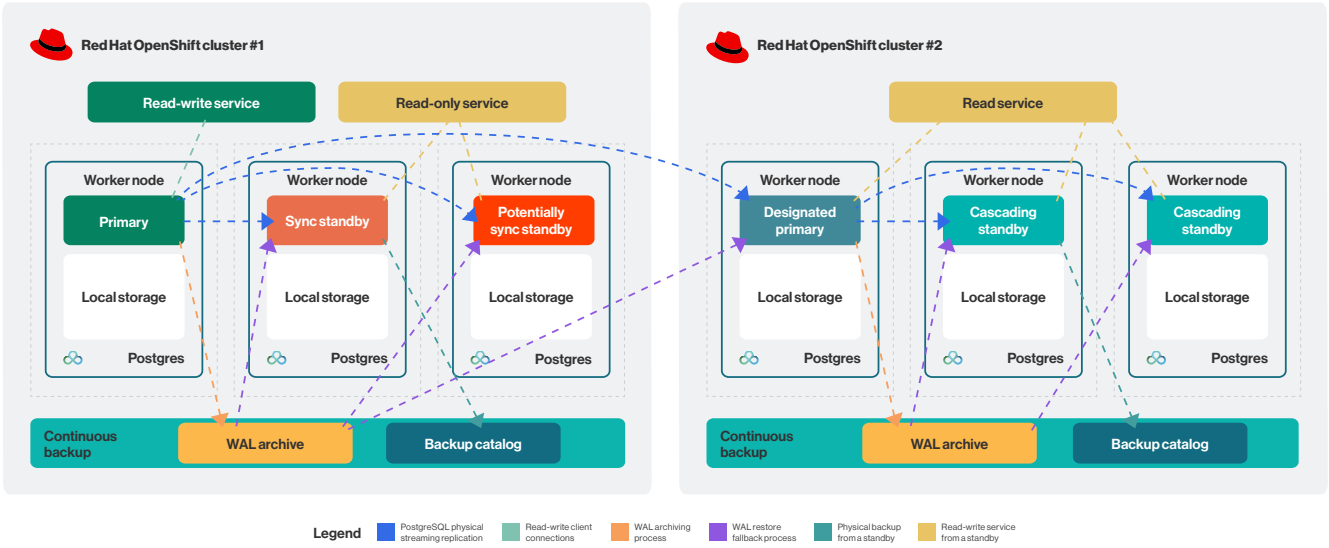

Figure 7. Example architecture of a Postgres cluster with cascading replication across two Red Hat OpenShift clusters

The goal of this document is to depict a single Red Hat OpenShift cluster as the SPoF for every PostgreSQL cluster managed by EDB PG4K running within it. In cloud environments, the SPoF is typically the region, thanks to availability zones. In most on-premises scenarios, the SPoF will be equivalent to a data center. In both cases, EDB PG4K can transparently and seamlessly handle both high availability (HA) and disaster recovery (DR) for every PostgreSQL cluster within the Red Hat OpenShift cluster. EDB PG4K can be configured with a backup strategy to deliver very low RTOs and guaranteed RPOs.

The most important recommendation is to dedicate worker nodes specifically for PostgreSQL workloads to ensure better performance predictability. Start with three worker nodes for PostgreSQL per Red Hat OpenShift cluster, and place taints on them to ensure that only PostgreSQL workloads managed by EDB PG4K can run. When the Red Hat OpenShift cluster cannot accommodate additional PostgreSQL clusters, you can add more worker nodes in multiples of three, using the same methodology based on taints.

Carefully evaluate the storage component, and consider using local storage directly mounted on each worker node, especially if you plan to use bare metal servers for PostgreSQL. In any case, benchmark both the storage and the database, following the documentation of EDB PGD4K.

For PostgreSQL, we recommend a three-instance cluster with a primary and two replicas configured with quorum-based synchronous replication of one replica. This setup ensures zero data loss for high availability. At a minimum, make sure that each instance is located on separate worker nodes via pod anti-affinity settings. This configuration also guarantees that node maintenance at the Red Hat OpenShift layer triggers rolling updates for each PostgreSQL cluster, minimizing downtime. This is called a shared-nothing architecture.

By using EDB PG4K, you achieve a highly integrated set of Postgres clusters within your Red Hat OpenShift infrastructure, including security, observability, logging, and more.

Once the Red Hat OpenShift cluster becomes the SPoF for each PostgreSQL cluster, you can extend HA and DR to cover additional Red Hat OpenShift clusters using EDB PG4K capabilities such as symmetric architectures, cascading replication, and most importantly, replica clusters for distributed topologies.

These capabilities are entirely declarative, making them suitable for infrastructure as code (IaC) across a fleet of Red Hat OpenShift clusters. This is critical for on-premises deployments of Red Hat OpenShift within a single data center—although in this case, EDB PG4K cannot autonomously control automated failover beyond the Red Hat OpenShift cluster, requiring external intervention.

For further discussion or questions, please contact the Sales team at enterprisedb.com/contact.

EDB