



# pgEdge Distributed PostgreSQL Buyers Guide User Guide

[Home](#) » [pgEdge](#) » pgEdge Distributed PostgreSQL Buyers Guide User Guide 

## pgEdge Distributed PostgreSQL Buyers



pgEdge Distributed PostgreSQL Buyers

### Contents

- [1 Instruction](#)
- [2 Key Questions](#)
- [3 Major Considerations](#)
- [4 Type of PostgreSQL Compatibility](#)
- [5 Support for new PostgreSQL versions](#)
- [6 Support for PostgreSQL extensions](#)
- [7 Implications of distributed database architecture on full PostgreSQL compatibility](#)
- [8 Advantages of the Broad PostgreSQL Community and Ecosystem](#)
- [9 Recommendations](#)
- [10 About the Author](#)
- [11 About pgEdge](#)
- [12 Customer Support](#)
- [13 Documents / Resources](#)
  - [13.1 References](#)

## Instruction

In the distributed database market many vendors today make claims about being “Postgres compatible” or “Postgres based”. This is no doubt a response to the growing and overwhelming popularity of Postgres with developers.

When contemplating adoption of a distributed Postgres database system careful consideration should be given to the level of effort required to migrate application code. Buyers should also look at the degree to which the product is outside the mainstream of the Postgres ecosystem, and the implications this raises.

## Key Questions

What is the basis of the vendor claims about being Postgres based or Postgres compatible? What level of effort will be involved to migrate existing applications and tooling to the new database system?

To what extent will we still be operating in and taking advantage of the extensive Postgres ecosystem if we adopt the vendor’s distributed database?

## Major Considerations

### Relationship with the PostgreSQL code base

In order to fully understand the basis of the vendor’s claims about Postgres it is highly advantageous to begin by investigating the product’s exact relationship with the PostgreSQL code base.

For open source products this is relatively straightforward: you can simply examine their source code in their Github repository. For a proprietary product you will need to ask the vendor to describe to you how their product is either built on Postgres, or how it delivers the claimed compatibility.



Whether the product is open source or not, the questions you need to ask here are:

- **(a)** Is the product’s core distributed functionality delivered by way of a standard extension (or extensions) to core Postgres itself?
- **(b)** Alternatively is the product a fork of the standard PostgreSQL code base, and if so how much does this

deviate from it (a “hard fork”)?

- **(c)** Does the product require any patches to be applied to standard PostgreSQL? Is the source code to these patches available?
- **(d)** Does the product incorporate PostgreSQL code by way of a straight copy of code into the product's own code base? If so, how recently has this been updated i.e. what version of Postgres was utilized in this way? Is this version still supported by the Postgres community? At least one vendor (Yugabyte) has based part of their product on a version of Postgres that has reached end-of-life status and is no longer supported by the community.
- **(e)** If the product is not making use of any Postgres code, how exactly is the claimed level of Postgres compatibility delivered?

plEdge Distributed PostgreSQL is built 100% on standard PostgreSQL, and the core distributed functionality is implemented via a standard Postgres extension called Spock. pgEdge Distributed PostgreSQL also includes several small (50 to 100 lines) optional patches to standard Postgres to implement certain advanced features. Source code is available at the pgEdge Github page for all components of pgEdge Distributed PostgreSQL.

## Type of Postgres Compatibility

The next consideration to look at is the type of Postgres compatibility that is claimed by the vendor. These claims fall into the following three categories.

**Wire Protocol Compatible** – Communication between application clients and the backend Postgres server take place over a TCP port (5432 by default). PostgreSQL has a well documented protocol for passing SQL commands and their results between the client and server. Certain distributed database products have adopted the Postgres wire protocol and this allows them to use the same language drivers as Postgres.

However, products that are only wire protocol compatible offer little in the way of true Postgres compatibility, since the SQL syntax and semantics supported will typically be quite different from standard Postgres. Postgres applications may be able to initially connect with the target database but very quickly will be exhibiting SQL syntax errors and unexpected behaviors.

**Syntax Compatible** – The next level of Postgres compatibility is SQL syntax compatibility. Does the product accept all Postgres SQL commands, or at least a subset, and execute them with the same semantics as standard Postgres? If syntax compatibility is claimed, is this compatibility with all current versions of Postgres, or just some earlier single version?

Particular attention should be paid as to whether the product supports Postgres functions and stored procedures, and if so does it support the same wide variety of programming languages that Postgres and various extensions do? This is a particularly important question for applications that make significant use of stored functions and procedures. Many corporations previously migrated their Oracle applications to Postgres, making use of the similarity in syntax between the native Oracle procedural language PL/SQL and its Postgres equivalent PL/pgSQL. In such cases full support for PL/pgSQL is likely a major requirement unless substantial redevelopment work is already contemplated.

Similarly attention should be given to support for triggers, which are rather widely used in Postgres applications.

Vendors claiming Postgres syntax compatibility will usually have a page on their website or in their documentation describing differences and missing features between their product's SQL implementation and that of standard Postgres. This should be examined to determine if these differences will create a meaningful amount of code migration overhead.

**Fully Postgres Based** – Finally we come to distributed databases that are fully based on standard PostgreSQL. Such products will typically package standard Postgres along with the extensions and patches required for distributed operation. This should be fully evident in the product’s code base,



These products are in all respects the same as stand-alone Postgres, with the same syntax, semantics and behaviors of standard Postgres, subject to limitations imposed by the distributed architecture of the product (see below). Hence it doesn’t make too much sense to call these products “Postgres compatible” when they are in fact fully based on Postgres.

Distributed Postgres products that are fully based on standard Postgres are able to take advantage of virtually all the extensions, tooling and add-on products available in the large and growing Postgres ecosystem. And just as importantly, application code in many cases does not require rework, although substantial testing is of course recommended.

pgEdge Distributed PostgreSQL is fully based on standard Postgres

**Support for new PostgreSQL versions**

The buyer should ask the vendor as to how soon support for new Postgres major versions is incorporated into the product. This typically should be no more than a few weeks or months from the annual Postgres major version release each September or October. However, because of the nature of how their products have been developed, some distributed database products do not add support for new major versions of Postgres. This means the buyer is not able to access the substantial innovation and improvements that go into each new Postgres major version release.

**PostgreSQL Supported Versions (green)**

PostgreSQL 18						
PostgreSQL 17						
<a href="#">master</a>						
PostgreSQL 16	2023-09-14	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">16.2</a> (2024-02-08)	<a href="#">REL_16_STABLE</a>	2028-11
PostgreSQL 15	2022-10-13	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">15.6</a> (2024-02-08)	<a href="#">REL_15_STABLE</a>	2027-11
PostgreSQL 14	2021-09-30	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">14.11</a> (2024-02-08)	<a href="#">REL_14_STABLE</a>	2026-11
PostgreSQL 13	2020-09-24	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">13.14</a> (2024-02-08)	<a href="#">REL_13_STABLE</a>	2025-11
PostgreSQL 12	2019-10-03	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">12.18</a> (2024-02-08)	<a href="#">REL_12_STABLE</a>	2024-11
PostgreSQL 11	2018-10-18	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">11.22</a> (2023-11-09)	<a href="#">REL_11_STABLE</a>	2023-11
PostgreSQL 10	2017-10-05	<a href="#">release notes</a>	<a href="#">announcement</a>	<a href="#">10.23</a> (2022-11-10)	<a href="#">REL_10_STABLE</a>	2022-11

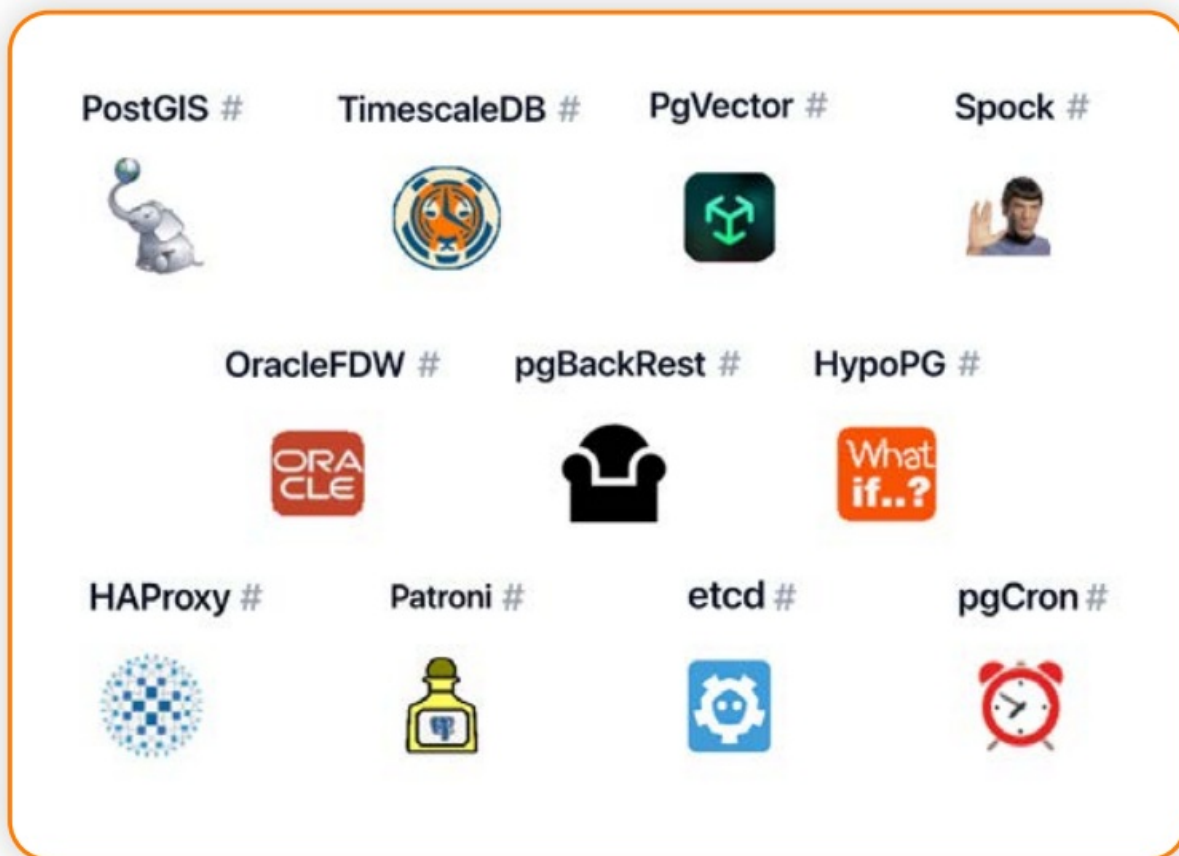
pgEdge delivers support for new major Postgres versions within a few weeks of each release. Additionally pgEdge

supports some of the beta releases ahead of the major release being available. (This is for evaluation and testing purposes only, we should add!)

## Support for PostgreSQL extensions

One of the major contributing factors to the success of Postgres in recent years has been its extensible architecture and the thousands of extensions that have been developed for it. In some cases these extensions allow Postgres to function as an entirely different type of database: for example, the PostGIS extension turns Postgres into a spatial database, and pgvector turns it into a vector database.

The buyer should determine the level of support in the product – either none, limited or full – for Postgres extensions. Does adding support for a particular extension require the product vendor to do work to create this support? Or do extensions just work without modifications to the extension or the product, as with standard Postgres?



pgEdge Distributed PostgreSQL supports the full range of available Postgres extensions, although users should test any needed extensions to ensure that they work correctly in a distributed multi-master configuration. pgEdge can provide a list of extensions that have already been tested and validated.

## Implications of distributed database architecture on full Postgres compatibility

Independent of their support for Postgres, distributed database systems broadly follow one of two different architectures:

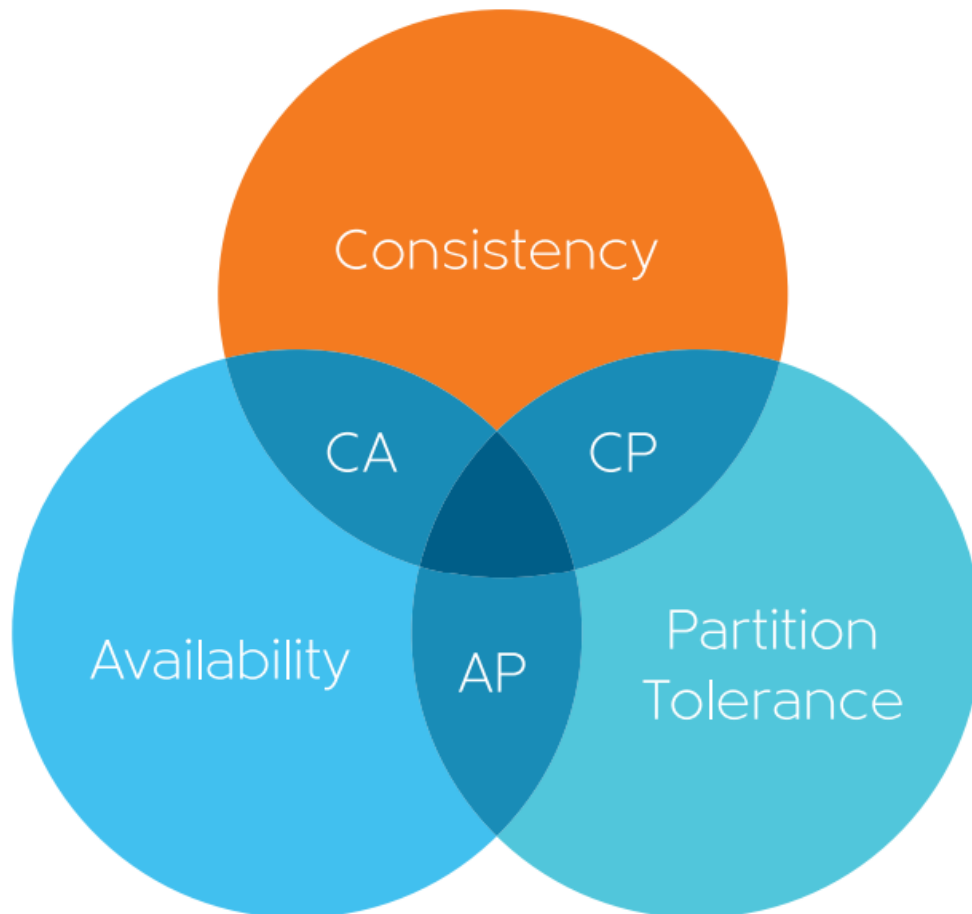
1. “CP” databases, prioritizing full consistency over availability and low latency
2. “AP” databases, prioritizing availability and low latency and providing eventual consistency.

(The “CP” and “AP” designations come from the CAP theorem, which you can read [here](#).)

CP Databases in the market currently include CockroachDB, YugabyteDB and Google Cloud Spanner. AP databases include pgEdge, EDB Postgres Distributed, Oracle Goldengate and numerous NoSQL databases such as MongoDB, Cassandra and DynamoDB.

Constraints of CP databases place limits on the Postgres features that can be supported by the products. Again, consult the vendor documentation for details on differences with standard Postgres.

Constraints of AP databases require that the application(s) can run in an eventually consistent model. The good news is that many can, as evidenced by the popularity of NoSQL AP databases such as MongoDB and Cassandra. It is also worth noting that at each node in a geographically distributed AP cluster, standard Postgres consistency, isolation levels and ACID compliance are seen among clients connecting to that node.



### **Advantages of the Broad Postgres Community and Ecosystem**

PostgreSQL development is driven by a vibrant developer community, the second largest in the world after Linux. This allows Postgres to continue to evolve and deliver a steady stream of innovation. Being able to tap into this community led development is one of the reasons organizations adopt Postgres.

If a distributed database product is not Postgres based and therefore not a part of the Postgres mainstream, and instead is driven by a single company (even if it is open source), organizations buying the product will no longer be able to benefit from the innovations coming from the Postgres community.





## Recommendations

Organizations evaluating Postgres compatible or Postgres based distributed database products should:

1. Assess the products based on the criteria outlined above.
2. Run a pilot project that includes migration of all or part of a key application to run on at least two of the shortlisted products. Look at the level of effort required to migrate the code used in the pilot, and use this to extrapolate to an overall level of migration effort.
3. For products that are not fully Postgres based, in addition to the migration lift, consider how important it is for the organization to be aligned with the Postgres community and ecosystem, and all the innovation, extensions, tooling and talent they provide

## About the Author



**Phillip Merrick**

Co-Founder/Chief Executive Officer pgEdge

Software developer by background, with multiple inventions in daily use by millions of people. Successful entrepreneur, technologist and CEO, specializing in enterprise software and cloud/SaaS markets in the USA, Europe and Asia/Pacific. Co-founder and/or CEO of webMethods, EDB, VisualCV, SparkPost, Fugue and pgEdge. Led several companies from initial startup phase through IPO and beyond, as well as three successful exits in the 9 to 10 figures range. Served in multiple public and board positions.

## About pgEdge


pgEdge's mission is to make it easy to build and deploy highly distributed database applications across the global network. Founded by industry veterans who have championed enterprise usage of the PostgreSQL database for several decades and helped run the world's largest managed database cloud services, pgEdge is headquartered in Northern Virginia. The founders have previously founded and/or led successful companies such as webMethods (NASDAQ: WEBM), EnterpriseDB (acquired by Bain Capital), SparkPost (acquired by MessageBird), OpenSCG (acquired by AWS) and Fugue (acquired by Snyk). Investors in pgEdge include Sands Capital Ventures, Grotech Ventures and Sand Hill East.

## Customer Support

201 N. Union Street  
Alexandria, VA 22314  
[www.pgedge.com](http://www.pgedge.com)



## Documents / Resources

	<p><a href="#">pgEdge Distributed PostgreSQL Buyers Guide</a> [pdf] User Guide Distributed Postgre SQL Buyers Guide, Postgre SQL Buyers Guide, Buyers Guide, Guide</p>
---	--

## References

- [pgEdge Fully Distributed PostgreSQL](#)
- [User Manual](#)

### Manuals+, Privacy Policy

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.