Microsoft

Introduction to Microsoft Azure Database for PostgreSQL

An overview of its features and capabilities

Abstract

Azure Database for PostgreSQL makes it easy to build new applications or migrate existing ones to the cloud using the programming languages and frameworks of your choice. As a fully managed database service, it increases your productivity by freeing you from database configuration and administration, operating system management, and all the other things you need to deal with when running PostgreSQL in a VM.

Azure Database for PostgreSQL is based on the latest community versions of PostgreSQL. It lets you continue using your favorite languages, tools, and extensions—like PHP, Python, Java, Node.js, Ruby on Rails, pgAdmin, pg_cron, and PostGIS. You can provision new databases in minutes and independently scale compute and storage in seconds, knowing you'll get built-in high availability, automatic backups and updates, and a service-level agreement of 99.99 percent.

With turn-key horizontal scalability, Azure Database for PostgreSQL can handle even the most demanding workloads. You'll also get advanced data security and compliance, including encryption at rest, encryption in transit, and built-in compliance with HIPAA, PCI DSS, FedRAMP, and more. Best of all, with Azure Database for PostgreSQL, you can pay as you go under a simple yet flexible pricing model, knowing that your database service is built on a world-class infrastructure with unmatched global reach.

Build or migrate your workloads with confidence, optimized for value

Highly available and flexible PostgreSQL

- Single-zone or zone-redundant high availability
- Up to 99.99% SLA
- AI-powered performance optimization

Advanced data security and compliance

- Encryption in motion and at rest
- Enhanced security with Azure Defender

Limitless scale with Hyperscale (Citus)

- High-performance, horizontal scalability
- Run anywhere with Azure Arc, including hybrid scenarios

The freedom of open-source with a simplified developer experience

- Fully compatible PostgreSQL
- Simplified, end-to-end deployment and low TCO
- Integrated with Azure Kubernetes Service and other Azure services

Intended audience

This paper is for architects, developers, and database administrators who are thinking of building and deploying new cloud apps on PostgreSQL or moving existing PostgreSQL apps to the cloud. It provides an overview of Azure Database for PostgreSQL and the value that it provides as a fully managed, open-source database service, including key use cases and an introduction to common application hosting environments. It also covers the various deployment options for Azure Database for PostgreSQL and introduces the key concepts you'll need to understand to best put each deployment option to use.

Contents

.

Introduction: PostgreSQL databases in the cloud

In the early days of computing, developers shared their code to learn from one another and evolve their software. This spirit of collaboration remains alive today with open-source software, which gives users rights to access, modify, and distribute source code—relying on a community of global contributors to achieve more than any one organization could on its own.

To fully appreciate the value of open-source software, it's worth acknowledging the many benefits it provides, including open standards that are accessible to all, full visibility into the code base, community efforts to improve and support the code, a lack of licensing fees, and lower development costs. Openly available source code also speeds the delivery of new offerings, helps prevent vendor lock-in, facilitates choice across platforms, and can help attract more and better talent. Best practices for maximizing performance and availability are shared freely, as are those for deployment and operations, and reliability is maximized through community vetting.

It's for all these reasons that open-source databases such as PostgreSQL, often called Postgres, have grown to be so popular. Since its initial release in 1989, PostgreSQL has earned a strong following due to its proven architecture, extensibility, reliability, and robust feature set, including support for custom data types and unstructured (JSON/JSONB) data. Unlike traditional relational databases, PostgreSQL has an object-oriented database model, in which objects, classes, and inheritance are natively supported in both database schemas and the query language. PostgreSQL is also extensible and can be used with many programming languages, including Python and Java, in addition to popular web app frameworks like Django and Ruby on Rails.

Due to its proven flexibility and reliability, PostgreSQL has become the open-source relational database of choice for many developers, earning it the title of DBMS of the Year from DB-Engines.com for three out of the past four years. In turn, the pervasiveness of PostgreSQL is fueling explosive growth in fully managed PostgreSQL database *services*, which let you reap the benefits of using PostgreSQL while avoiding the many pitfalls of on-premises solutions or hosted, self-managed virtual machines. For example, with platform-as-a-service (PaaS) cloud services, you won't need to deal with deployment efforts, continual patching, painstaking performance optimization, complex high-availability mechanisms, intricate security and compliance concerns, and continual worries about scalability—all of which can lead to unpredictable DBA workloads, delays, and costs.

To make PostgreSQL work in the enterprise, however, you'll need a lot more than just a hosted PostgreSQL database service. Rarely does any database exist in isolation, which is why you'll want to make sure your chosen platform can deliver everything else you'll need. Specific services will depend on your app, such as whether you'll be building a containerized application, implementing microservices, ingesting massive data volumes, implementing real-time streaming analytics, or building an AI-powered chatbot. And don't forget about development tools, software frameworks, and integration; you won't want to complicate things unnecessarily in those areas either. After all, even if a cloud platform provides all the services you need, it won't do you much good if you need to learn an entirely new skill set to best put those services to use.

In the rest of this paper, we provide an overview of Microsoft Azure Database for PostgreSQL, a fully managed PostgreSQL database service, and the value that it brings to you as an architect, developer, or database administrator. We'll also cover the different deployment options for Azure Database for PostgreSQL and the key concepts you'll need to understand to best put each deployment option to use. Finally, we'll examine some popular use cases for Azure Database for PostgreSQL and some application hosting environments that complement it.

Azure Database for PostgreSQL

Azure Database for PostgreSQL brings together PostgreSQL innovations with all the benefits of the Microsoft Azure cloud. As a fully managed service, it frees you from the burden of infrastructure and database management to help you easily build new applications or migrate existing ones to the cloud using the programming languages and frameworks of your choice. You can provision new databases in minutes and independently scale compute and storage, knowing that you won't need to pay extra for things like automatic database patching and backups, monitoring tools, AI-powered performance optimization, or essential security features.

Azure Database for PostgreSQL gives you:

- A highly available and flexible PostgreSQL service that automatically handles all maintenance, patching, and updates to let you focus on application innovation.
- Support for the latest community versions of PostgreSQL, including access to popular extensions and support for popular frameworks like Ruby on Rails, Python with Django, Java with Spring Boot, and Node.js.
- A simplified developer experience, including streamlined deployment via tight integration with Azure App Service, Azure Kubernetes Service, and GitHub Actions.
- A 99.99-percent, financially-backed service level agreement.
- Advanced data security and compliance features, including double encryption at rest, encryption in transit, Azure Defender[1], and built-in compliance with HIPAA, PCI DSS, FedRAMP, and more.
- Multiple pricing options—including the ability to pay-as-you-go or to save money by paying for compute capacity in advance.

Our new Flexible Server deployment option for Azure Database for PostgreSQL delivers additional innovations, including greater flexibility and more granular control over database configuration and management. Additional capabilities in Flexible Server include custom maintenance windows, multiple configuration parameters for fine-grained tuning, zone-redundant high availability, and additional cost optimization controls—including burstable compute and the ability to start and stop servers to save money.

Last but not least, our Hyperscale (Citus) deployment option delivers virtually limitless scalability and performance, including a basic pricing tier that makes it easy to get started with lower costs and then scale out later. Hyperscale (Citus) server groups can even be deployed as an Azure Arc-enabled data service, which uses Kubernetes to let you run Azure Database for PostgreSQL outside of Azure, on the infrastructure of your choice.

The capabilities provided by Azure Database for PostgreSQL make it ideal for hosting enterprise-scale production workloads. You can get started immediately and build a broad range of solutions, knowing that you won't be limited by infrastructure issues, scalability, availability concerns, geographic presence, or excessive upfront costs. The possibilities are endless, which is one reason why open-source database services are becoming so popular.

As an Azure data service, Azure Database for PostgreSQL is backed by Azure compute and Azure Storage. And it's built to work with the extensive range of other Azure services—including those for data ingestion,

[1] Azure Defender is a separate Azure service and is available at an additional cost.

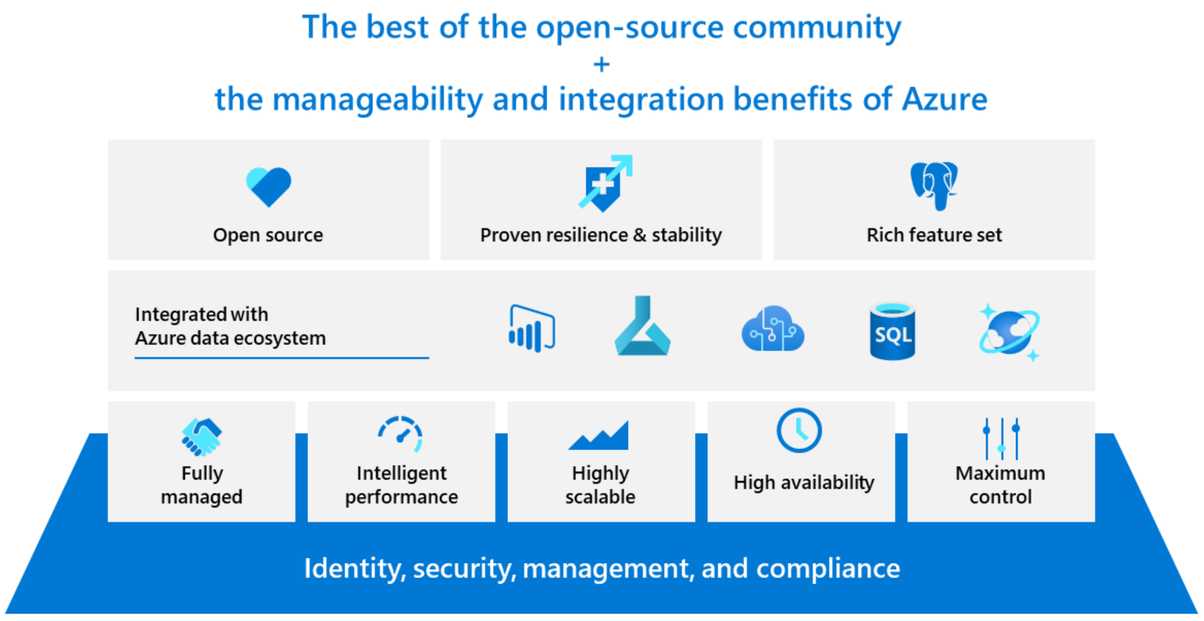
integration, application hosting, AI and machine learning, analytics, visualization, data governance, DevOps, and more.

Like all Azure services, Azure Database for PostgreSQL is built on a world-class infrastructure with unmatched global reach; Azure has more global regions than any other cloud provider, enabling you to bring your apps closer to your users around the world, preserve data residency, and satisfy compliance demands. Finally, when you choose Azure Database for PostgreSQL, you'll get comprehensive compliance and Azure IP Advantage, a program for Azure customers that provides protection against intellectual property (IP) risks.

Microsoft is an active participant in the PostgreSQL community, with more top-level PostgreSQL committers than any other public cloud provider. In addition to working on Azure Database for PostgreSQL, the PostgreSQL team at Microsoft is actively involved in giving back to the PostgreSQL community, such as through the team's recent work to speed-up recovery and VACUUM in PostgreSQL 14.

In the rest of this paper, we'll cover:

- The three deployment options for Azure Database for PostgreSQL and when you might want to consider using each one.
- The key concepts you'll need to understand for each deployment option: Single Server, Flexible Server, and Hyperscale (Citus).
- Popular use cases for Azure Database for PostgreSQL.
- Commonly used application hosting environments and other services for building apps with Azure Database for PostgreSQL.


The best of the open-source community
+
the manageability and integration benefits of Azure
Open source
Proven resilience & stability
Rich feature set
Integrated with Azure data ecosystem
SQL
Fully managed
Intelligent performance
Highly scalable
High availability
Maximum control
Identity, security, management, and compliance

Deployment options

With Azure, you can run PostgreSQL in two ways: within a hosted VM infrastructure-as-a-service (IaaS) environment, or within a fully managed platform-as-a-service (PaaS) environment. Although this paper focuses on the range of PaaS options provided on Azure, to understand the value provided by those options, it's worth first taking a quick look at you get—and what you don't get—when running a self-managed instance of PostgreSQL within a VM.

With IaaS, you have full control over the PostgreSQL database engine running within a VM, along with the underlying operating system, tools, and utilities—including the ability to run whichever versions you choose. However, this comes at a cost, in that you'll need to handle all database and operating system management and administration tasks on your own: maintaining and patching your servers, high availability design, disaster recovery, security design, performance tuning, and more. These tasks can require substantial time and effort, contributing significantly to your overall total cost of ownership (TCO).

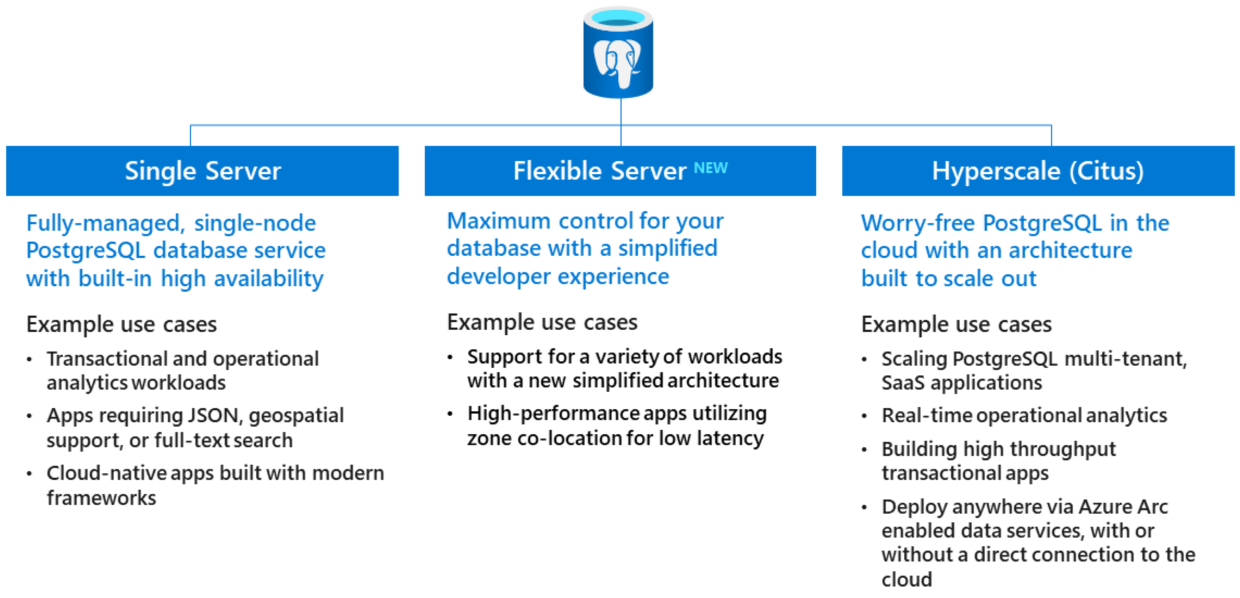
Azure Database for PostgreSQL, on the other hand, is a *fully managed* PaaS database service. This means that Microsoft automatically configures and manages the underlying infrastructure, operating system, and database software so that you don't have to—including the application of patches and security fixes. You'll also have access to a wealth of features that aren't typically included out-of-the-box with an IaaS environment, such as automated backups, built-in security and performance features, on-demand scalability, and guaranteed high availability.

However, even within the PaaS domain, one size rarely fits all. That's why Azure Database for PostgreSQL supports three different deployment options:

- **Azure Database for PostgreSQL - Single Server,** the first deployment option we introduced, automatically handles most database management functions, including patching, backups, and security—with minimal user configuration and control.
- **Azure Database for PostgreSQL - Flexible Server** provides new innovations that go beyond what's in Single Server, including greater flexibility and more granular control over database configuration and management. Flexible Server also provides additional cost optimization controls, including a burstable compute pricing tier and the ability to start and stop the server for development or test scenarios—and only pay for storage while the database is stopped. Flexible Server is recommended for most new Azure Database for PostgreSQL deployments, including migrations from Oracle, migrations from VM-based (IaaS) PostgreSQL environments, and most new cloud-native apps.
- **Azure Database for PostgreSQL - Hyperscale (Citus)** is designed for apps that require superior scalability and performance. Hyperscale (Citus) deployments automatically scale out across multiple PostgreSQL database servers using sharding, employing query parallelization for faster responses on large datasets—all using standard connection libraries, with minimal (if any) required changes to your app. In addition to the general pricing tier for Hyperscale (Citus), which can scale out to dozens of worker nodes, there's a basic pricing tier that lets your run an entire Hyperscale (Citus) deployment on just one node—providing a simple and cost-effective way to start with a small server group and scale out later. The Hyperscale (Citus) deployment option is well suited for multi-tenant apps, real-time operational analytics, and high-throughput transactional workloads.

Choose the right PostgreSQL server option in Azure provides additional information on the different ways you can deploy PostgreSQL on Azure, including the main differences between each option and how specific business motivations might influence whether to choose IaaS or PaaS.

The following chapters in this guide cover the above deployment options in greater detail, including the key concepts you'll need to understand to best put each deployment option to use.



Single Server

Fully-managed, single-node PostgreSQL database service with built-in high availability

Example use cases

- Transactional and operational analytics workloads
- Apps requiring JSON, geospatial support, or full-text search
- Cloud-native apps built with modern frameworks

Flexible Server NEW

Maximum control for your database with a simplified developer experience

Example use cases

- Support for a variety of workloads with a new simplified architecture
- High-performance apps utilizing zone co-location for low latency

Hyperscale (Citus)

Worry-free PostgreSQL in the cloud with an architecture built to scale out

Example use cases

- Scaling PostgreSQL multi-tenant, SaaS applications
- Real-time operational analytics
- Building high throughput transactional apps
- Deploy anywhere via Azure Arc enabled data services, with or without a direct connection to the cloud

Preview deployment options and features

Throughout this guide, we've included features and capabilities that are in preview—meaning that they're available for early access but have yet to reach general availability. In general, preview features are provided without a service level agreement, might have constrained capabilities, and should be tested thoroughly before using them to support production workloads.

Preview features change regularly, when our engineering and product teams are confident that they're ready to be promoted to general availability. Our online documentation for Azure Database for PostgreSQL is the definitive source of what's currently in preview. Azure updates is another good source to monitor—you'll typically see an entry when a preview feature is introduced and another entry when it reaches general availability. Finally, the Azure Database for PostgreSQL blog often has posts from our engineering team about new features and capabilities in Azure Database for PostgreSQL—including guidance on how to put them to use.

Key concepts: Single Server

Following are some of the key concepts you'll need to understand to get the most out of the Azure Database for PostgreSQL - Single Server deployment option, which automatically handles most database management functions and is built to deliver 99.99% availability.

Azure Database for PostgreSQL - Single Server provides a deeper overview of the Single Server deployment option.

Supported versions

Microsoft's goal is to have Azure Database for PostgreSQL- Single Server support the current major version of the PostgreSQL engine and the two prior major versions. As of June 2021, the Single Server deployment option supports the following versions:

- PostgreSQL Version 11
- PostgreSQL Version 10
- PostgreSQL Version 9.6

Major enhancements in PostgreSQL 11 include improvements associated with partitioning, parallelism, stored procedures, Windows function support, and covering indexes, as well as several other performance enhancements. The PostgreSQL 11 release notes provide more information about the new functionality in PostgreSQL 11 and how it differs from PostgreSQL 10.

Supported versions will change over time, so it's best to check the online documentation to see which PostgreSQL versions are currently supported for Azure Database for PostgreSQL - Single Server.

Note: Starting with PostgreSQL 10, a major version is indicated by increasing the first part of the version—for example, 10 to 11. Before PostgreSQL 10, a major version was indicated by increasing either the first or second part of the version number, such as 9.5 to 9.6.

Updates and upgrades

Azure Database for PostgreSQL - Single Server automatically manages patching for minor version updates. Currently, major version upgrades aren't supported. For example, upgrading from PostgreSQL 10 to PostgreSQL 11 isn't supported. If you want to upgrade to the next major version, you can use the Azure Database Migration Service or perform a dump and restore.

Connection libraries

Most language and client libraries used to connect to PostgreSQL server are external projects and are distributed independently. For a list of the libraries and drivers that you can use to build applications that connect to and query Azure Database for PostgreSQL - Single Server, see Connection libraries for Azure Database for PostgreSQL - Single Server.

Regardless of which tools you choose, remember that you'll need to think about connection resiliency and how to handle transient errors.

PostgreSQL extensions

One of the great things about PostgreSQL is that it supports extensions, which let you extend the functionality of your database. The PostGIS extension is a great example—it's why PostgreSQL is the world's most popular open-source geospatial database. Other popular extensions include pg_cron (for scheduling cron jobs on PostgreSQL), pg_trgm (for fuzzy string matching), xml2 (for parsing and querying xml objects), and orafce (which lets you use Oracle functions and packages within PostgreSQL). After you load them, extensions can function like built-in features.

When you deploy Azure Database for PostgreSQL - Single Server, the pg_stat extension is automatically installed, enabling you to track execution statistics for SQL statements. PostgreSQL extensions in Azure Database for PostgreSQL - Single Server provides more information on extensions, including how to load them and a list of supported extensions.

Databases and servers

Azure Database for PostgreSQL - Single Server exposes access and features at the *server* level, which is created within an Azure subscription and is the parent resource for *one or more individual databases*. It collocates the computing resources for those databases in a specified Azure region, provides a namespace for the databases, and serves as a connection endpoint for server and database access. A server also defines the scope for management policies that apply to those databases—including sign-in, firewall, users, roles, and configurations. The online documentation for Azure Database for PostgreSQL - Single Server provides more information on the concepts of databases and servers.

You can create and manage servers by using the Azure portal, the Azure CLI, Azure PowerShell, or Azure Resource Manager templates. Which to use depends on your skill-set and preferences. In general:

- Linux admins and traditional open-source developers often prefer a command-line interface, with the Azure CLI providing just that. The Azure CLI is also a good choice for developers and Linux admins who rely heavily on Ruby scripts, Unix shell environments, Groovy, and so on, and want to reuse such scripts when managing PostgreSQL in a PaaS environment.
- Azure PowerShell provides similar scripting capabilities for developers and admins who are familiar with PowerShell, with virtually no additional learning curve required.
- Azure portal is often a preferred option for developers or admins from all backgrounds who want the power and productivity of a GUI, including ready access to built-in monitoring capabilities and performance-tuning aids like Query Store, Query Performance Insights, and Performance Recommendations. Azure portal also provides access to Azure Cloud Shell, an interactive, authenticated, browser-accessible shell for managing Azure resources. You can use either Linux Bash or PowerShell with Azure Cloud Shell, enabling you to choose the shell experience that best suits the way you work.
- Azure Resource Manager (ARM) templates enable you to define your infrastructure as code, using JavaScript Object Notation (JSON) to specify the resources to deploy—such as an Azure Database for PostgreSQL instance—and the properties for those resources. The infrastructure code becomes part of your project and can be stored and versioned in your source code repository, enabling you to automate the deployment of your infrastructure as well as the code it runs.

Server pricing tiers and resources

Under the Single Server deployment option, pricing for an Azure Database for PostgreSQL server is determined by the configuration of resources that you specify for that server, including the pricing tier, number of vCores, and amount of storage. Create a single database under the server, and that database has exclusive access to those resources; create multiple databases under the server, and those resources are shared.

Pricing tiers

You can create a server in one of three pricing tiers, which are differentiated by the amount of compute (vCores) that can be provisioned, the amount of memory per vCore, and the technology used for data storage. In general:

- **Basic** is good for workloads that require only light compute and I/O—such as test servers and small-scale, infrequently used applications.
- **General purpose** is good for workloads that require balanced compute and memory, with scalable I/O—including web apps, mobile apps, and many other enterprise apps. This is our most popular option and works well for most scenarios.
- **Memory optimized** is good for workloads that require in-memory performance for faster transaction processing and greater concurrency—including real-time data processing and high-performance transactional or analytical apps.

After you create a server, you can adjust the number of vCores, the hardware generation, and the pricing tier up or down, within seconds. Similarly, you can increase the amount of storage or adjust your backup retention period up or down, without application downtime.

Note: Changing to and from the basic tier is not supported, so if you're planning to scale, we recommend you start with the general-purpose or memory-optimized tiers. Also, you can't change the backup storage type after a server is created.

Finally, it's worth pointing out that each option within each pricing tier has an associated set of limits, such as the maximum number of concurrent connections.

Reserved capacity

Azure Database for PostgreSQL - Single Server supports reserved capacity pricing, which lets you reserve compute power for a period of one year or three years. By taking advantage of it, you can save up to 60 percent compared to the regular, pay-as-you-go payment options.

Prepay for Azure Database for PostgreSQL - Single Server compute resources with reserved capacity provides more information on reserved capacity, including how to determine the right server size and how to purchase reserved capacity. A pricing calculator can be found on the Azure Database for PostgreSQL pricing page.

Storage

When you provision a server, you'll also need to provision the amount of storage capacity available to it. This storage is used for the database files, temporary files, transaction logs, and PostgreSQL server logs. The total storage you provision also defines the I/O capacity available to your server in terms of I/O operations per second (IOPS). Azure Database for PostgreSQL – Single currently supports up to 16 TB of storage and up to 20,000 IOPS.

You can add additional storage capacity during and after the creation of the server. You can also enable storage auto-grow, which allows the system to grow storage automatically based on the storage consumption of your workload.

Azure Database for PostgreSQL - Single Server provides up to 100 percent of your provisioned server storage as backup storage, at no additional cost. Any backup storage you use in excess of this amount is charged in GB per month, according to the service's pricing model.

Security

With Azure Database for PostgreSQL, you have several capabilities to help ensure safe and secure access. To help protect your data, it's encrypted both in-transit and at-rest. Additional security mechanisms are built into the networking and access management mechanisms for Azure Database for PostgreSQL, and you can opt into Azure Defender for an even greater level of security. Security in Azure Database for PostgreSQL - Single Server provides a deeper introduction to these security features.

Configuring SSL

Enforcing Secure Sockets Layer (SSL) connections between your client applications and an Azure Database for PostgreSQL server encrypts all traffic between the two, which can help protect against "man-in-the-middle" attacks. Enforcement of SSL connections is enabled by default when a new Azure Database for PostgreSQL server is provisioned. It's worth noting, however, that some application frameworks don't enable SSL by default during installation, so you'll want to check this if your app is failing to connect to your database server.

The online documentation for Azure Database for PostgreSQL provides more information on configuring SSL.

Threat protection with Azure Defender

Azure Defender for open-source relational databases can help detect anomalous activities that may indicate unusual or potentially harmful attempts to access or exploit your databases. It's part of the integrated cloud workload protection platform within Azure Security Center, a unified package of advanced security capabilities, which is available at an additional cost.

When Azure Defender detects a threat in any area of your environment, it generates a security alert that describes details of the affected resources, suggested remediation steps, and, in some cases, an option to trigger a logic app in response. Alerts are triggered upon anomalous database access and query patterns, suspicious database activities, and brute-force attacks. The alerts reference page provides a full list of security alerts for Azure Database for PostgreSQL and other open-source relational databases.

Introduction to Azure Defender for open-source relational databases provides more information on its capabilities and benefits. Our quickstart on enabling Azure Defender walks you through how to set it up.

Authenticating with Azure Active Directory

You can connect to Azure Database for PostgreSQL - Single Server using Azure Active Directory (Azure AD) authentication, using identities defined in Azure AD. This lets you manage database user identities in a central location, alongside user identities for other Azure services, thus simplifying and centralizing permission management. Azure AD supports federation with your on-premises Active Directory infrastructure to enable single sign-on (SSO) to cloud applications and data services.

Azure Active Directory authentication with PostgreSQL covers the benefits of this approach and how it works, including administrator types, permissions, supported connection methods, and additional considerations.

Data encryption with customer managed keys

By default, Azure Database for PostgreSQL leverages Azure Storage encryption to encrypt data at-rest using Microsoft-managed cryptographic keys. If you'd prefer, you can use your own keys, enabling your organization to implement separation of duties in the management of keys and data. When you choose to use your own keys, you'll be responsible for—and in full control of—the key's lifecycle, key usage permissions, and auditing of operations on keys.

Azure Database for PostgreSQL - Single Server data encryption with a customer-managed key provides more information on the benefits of using your own keys, requirements for doing so, and how the process works.

Customer managed keys require the use of Azure Key Vault, which is free for up to 12 months and 10,000 transactions with a new Azure free account.

Infrastructure double encryption

With Azure Database for PostgreSQL, data (including backups) are always encrypted on disk using a FIPS 140-2 validated cryptographic module and an AES 256-bit cipher. Infrastructure double encryption provides additional data protection through a second layer of encryption, which uses a different FIPS 140-2 validated cryptographic module with a different encryption algorithm and a second, service-managed cryptographic key. Infrastructure double encryption is not enabled by default because the additional layer of encryption may have a performance impact.

Azure Database for PostgreSQL infrastructure double encryption provides more information on the benefits of infrastructure double encryption, supported scenarios, and limitations.

Security controls by Azure policy

Regulatory Compliance in Azure Policy provides initiative definitions that are created and managed by Microsoft—known as built-ins—for the compliance domains and security controls related to different compliance standards. You can assign the built-ins for a security control individually to help make Azure resources compliant with a specific standard. Azure Policy Regulatory Compliance controls for Azure Database for PostgreSQL lists the compliance domains and security controls for Azure Database for PostgreSQL.

Security baseline

The Azure Security Benchmark contains recommendations to help you improve the security of your applications and data on Azure. It focuses on cloud-centric control areas. These controls are consistent with well-known security benchmarks, such as those described by the Center for Internet Security (CIS) Controls Version 7.1.

Azure security baseline for Azure Database for PostgreSQL - Single Server applies guidance from the Azure Security Benchmark version 1.0 to Azure Database for PostgreSQL - Single Server. Topics are grouped by the security controls defined by the Azure Security Benchmark, including network security, logging and monitoring, identity and access control, data protection, vulnerability management, inventory and asset management, secure configuration, malware defense, data recovery, incident response, penetration tests, and red team exercises—with links to the Azure Security Benchmark documentation for more information on each topic.

Networking

Connections to Azure Database for PostgreSQL - Single Server are established through a gateway that routes incoming connections to the physical location of your server in an Azure server cluster. As a client connects to the database, the connection string to the server resolves to the gateway IP address. Inside the database cluster, traffic is forwarded to appropriate Azure Database for PostgreSQL server.

Under this architecture, in order to connect to Azure Database for PostgreSQL - Single Server from outside of Azure, you may need to open up your client-side firewall to allow outbound traffic initiated from the client to reach our gateways. Connectivity architecture in Azure Database for PostgreSQL provides additional information on this topic, including a complete, per-region list of the IP addresses used by our gateways and the answers to some frequently asked questions.

Firewall rules

By default, all access to an Azure Database for PostgreSQL server is blocked by its firewall. You can use firewall rules to specify which computers can access a server, based on the originating IP address for each request. These firewall rules apply to all databases on the logical server and are configured as ranges of IP addresses. You can also configure the firewall to allow access from all Azure services and all Azure subnets. Firewall rules can be configured by using the Azure portal or the Azure CLI.

The online documentation for Azure Database for PostgreSQL - Single Server provides more information about firewall rules, including connecting from the internet, connecting from Azure, connecting from a virtual network, programmatically managing firewall rules, and troubleshooting firewall issues.

Virtual network rules

Virtual network (VNet) rules, a firewall security feature, control whether your Azure Database for PostgreSQL server accepts communications sent from particular subnets on Azure virtual networks. To create a virtual network rule, you must first have a VNet and a VNet service endpoint to reference. If you configure your database firewall to allow access from all Azure services and all Azure subnets, you can use virtual network rules to achieve much more granular control.

The online documentation for Azure Database for PostgreSQL provides more information about VNet service endpoints and virtual network rules, including their benefits, characteristics, limitations, and use with ExpressRoute.

Private Link

Azure Private Link allows you to create private endpoints for Azure Database for PostgreSQL - Single Server to bring it inside your VNet. The private endpoint exposes a private IP address within a subnet, which you can use to connect to your database server just like any other resource in the VNet. You can use Private Link to establish cross-premises access to a private endpoint using ExpressRoute, private peering, or VPN tunneling—or choose to disable all access via a public endpoint. Private Link for Azure Database for PostgreSQL - Single Server discusses relevant use cases and how to use Private Link with Azure Database for PostgreSQL.

Planned maintenance

Azure Database for PostgreSQL - Single Server performs automated patching of the underlying hardware, OS, and database engine—including new service features as well as security and software updates. No user action or configuration settings are required for patching, and patches are tested extensively before they're rolled out using safe deployment practices. Minor version upgrades for the PostgreSQL engine are automatic and are included as part of the patching cycle.

A planned maintenance event is a maintenance window when such service updates are deployed to servers in a given Azure region. At a minimum, they're 30 days apart. You'll receive a notification event when a service update is deployed in the region hosting your servers, including a notification of the next maintenance window 72 hours in advance.

Planned maintenance notification in Azure Database for PostgreSQL - Single Server provides more information on this topic, including duration and customer impact, how to receive and view planned maintenance notifications, and other useful information.

Business continuity

Azure Database for PostgreSQL - Single Server provides several features that you can use to help ensure business continuity and prevent data loss. Overview of business continuity with Azure Database for PostgreSQL - Single Server introduces these business continuity features and how you can put them to use, including an overview of recovery scenarios and how they can affect your Recovery Time Objective and Recovery Point Objective.

High availability

Azure Database for PostgreSQL - Single Server provides guaranteed high availability to help you avoid application downtime—including a financially backed service-level agreement of 99.99 percent for all services that are in general availability. This high-availability model is based on built-in failover mechanisms that are triggered when a node-level interruption occurs, such as in the event of a hardware failure.

The entire failover process typically takes tens of seconds and works as follows: At all times, changes to a server occur in the context of a transaction, with changes recorded synchronously in Azure storage when the transaction is committed. If a node-level interruption occurs, the database server automatically creates a new node and attaches data storage to the new node. Any active connections are dropped, any in-flight transactions aren't committed, and an internal gateway transparently redirects any new connections to the new instance.

Because of how the failover process works, it's important that you build your database applications to detect and retry dropped connections and failed transactions. When your application retries, its connection is transparently redirected to the newly created instance. And because the redirect is handled transparently by the internal Azure gateway, your client application's connection string remains the same.

The same failover model makes it easy to scale compute resources up or down. When you do so, a new compute instance with the specified resources is created, then existing data storage is detached from the original instance and attached to the new instance. Client applications are disconnected, open, uncommitted transactions are canceled, and, after the client application retries the connection or makes a new connection, the gateway directs the connection to the newly sized instance.

The online documentation for Azure Database for PostgreSQL - Single Server provides more information on high availability.

Backup and restore

Azure Database for PostgreSQL supports locally redundant backups in the basic tier, and both locally redundant and geographically redundant backups in the general-purpose and memory-optimized tiers. You can take advantage of these features to provide business continuity, such as recovering a server after a user or application error, or in the unlikely event of an Azure regional datacenter outage.

To understand how you can use backup and restore as part of a comprehensive business continuity strategy, it's worth taking a closer look at how these features work: Azure Database for PostgreSQL automatically creates full, differential, and transaction-log backups, which you can use to restore a server to any point in time within your configured backup retention period. Generally, for servers with up to 4TB of storage, full backups occur weekly, differential backups occur twice a day, and transaction-log backups occur every five minutes.

In the subset of Azure regions that support up to 16TB or storage, backups are snapshot-based. The first full snapshot backup is scheduled immediately after a server is created and is retained as the server's base backup. Subsequent snapshot backups are differential backups only and, while they do not occur on a fixed schedule, are performed three times per day. Transaction log backups occur every five minutes.

The default backup retention period is 7 days and can be configured for up to 35 days. All backups are secured using AES 256-bit encryption.

Backup storage options

The basic pricing tier offers only locally redundant backup storage, whereas the general-purpose and memory-optimized pricing tiers provide the option to augment locally redundant backups with geographically redundant (geo-redundant) backup storage. When you choose geo-redundant backup storage, your backups aren't just stored within the region in which your server is hosted. They're also replicated to a paired datacenter in another region within the same geography, allowing you to restore your server in any Azure region in the event of a disaster. There's a delay between when backups are taken and when they're replicated to another region, so, if a disaster occurs, you may face some data loss.

When you provision a server, you need to decide whether to augment locally redundant backups with geo-redundant backups because you can't change this option afterward. Whichever option you choose, Azure Database for PostgreSQL provides up to 100 percent of your provisioned server storage as backup storage at no additional cost.

Performing a restore creates a new server from the original server's backups. Recovery time depends on several factors, including the size of the database or databases, transaction log size, network bandwidth, and the total number of databases being recovered in the same region at the same time. The estimated recovery time is usually less than 12 hours.

Point-in-time restore is available with either backup option. It creates a new server in the same region as your original server. Geo-restore, which is available only if you configured your server for geo-redundant storage, lets you restore your server to a different region.

The online documentation for Azure Database for PostgreSQL - Single Server provides additional detail on how backup and restore work.

Long-term backup retention

Native backup features within Azure Database for PostgreSQL - Single Server are well-suited for operational recoveries. They can be combined with the long-term retention capabilities provided by Azure Backup to extend your backup retention period for up to 10 years, as may be needed to help meet compliance needs. Azure Database for PostgreSQL backup with long-term retention provides an overview of how Azure Database for PostgreSQL can be used with Azure Backup.

Monitoring and tuning

Monitoring your servers not only helps you know when troubleshooting might be needed, but it also gives you the data needed for performance tuning. Azure Database for PostgreSQL - Single Server provides several built-in tools for monitoring and tuning, including resource usage metrics that are captured at one-minute intervals and stored for 93 days. You can also enable logging on your server, and then send those logs to Azure Monitor, Event Hubs, or an Azure Storage account. In addition, Azure Database for PostgreSQL - Single Server provides features that let you track query performance over time, visualize that data in the Azure portal, and identify opportunities to improve workload performance.

The online documentation for Azure Database for PostgreSQL provides more information about the monitoring and tuning features in Azure Database for PostgreSQL - Single Server.

Logging and audit logs

Azure Database for PostgreSQL - Single Server lets you configure and access standard PostgreSQL logs, which can be used to help identify, troubleshoot, and repair configuration errors and performance issues.

Types of data you can log include errors, query information, auto-vacuum records, connections, and checkpoints.

Logs in Azure Database for PostgreSQL - Single Server covers how to configure logging on your server, access log files, and configure diagnostic settings to send your PostgreSQL logs to Azure Monitor, Event Hubs, and Azure Storage. Audit logging in Azure Database for PostgreSQL - Single Server covers how to configure detailed session and object audit logging, as enabled by the pgAudit extension for PostgreSQL.

Intelligent performance optimization

Azure Database for PostgreSQL – Single Server provides intelligent performance optimization, a set of features that make it easy to optimize query performance.

- Query Store tracks query performance over time, including query runtime statistics and wait events. You can control data collection and storage via various configuration settings.
- Query Performance Insight works with Query Store to provide visualizations within the Azure portal, which you can use to identify key queries that impact performance.
- Performance Recommendations identifies opportunities for creating new indexes that may improve workload performance. It takes into consideration various database characteristics, including the database's schema and the data collected by Query Store.

Azure Advisor recommendations

The Azure Advisor system uses telemetry to issue performance and reliability recommendations for Azure Database for PostgreSQL - Single Server, which are made available in the Azure portal. In general, there are three types of recommendations:

- Performance recommendations – based on CPU usage, memory pressure connection pooling, disk utilization, and other server parameters.
- Reliability recommendations – based on storage limits and connection limits.
- Cost optimization – including recommendations on server right-sizing.

When recommendations are available, a preview will appear as a banner notification in the Azure portal, under Overview. Details can be viewed in the Notifications section, just below the resource usage graphs. The online documentation provides more information on Azure Advisor for Azure Database for PostgreSQL.

Replication

Read replicas and logical decoding, which are both forms of replication, can be used to capture data changes in PostgreSQL and send them to another system. Read replication is useful for workloads that have a read-only component, enabling that workload component to be offloaded from the master (read/write) server to a read-only replica. Logical replication works at a different level, enabling a PostgreSQL server to send a stream of data modifications to one or more external consumers. Both read replicas and logical decoding rely on the PostgreSQL write ahead log for information, with logical decoding requiring a higher level of logging than read replicas.

Read replicas

Azure Database for PostgreSQL - Single Server supports read replicas, which you can use to replicate data from an Azure Database for PostgreSQL server (master) to up to five read-only servers (replicas). Read replicas can help improve the performance and scale of read-intensive workloads, which can be isolated to the replicas while write workloads are directed to the master. A common scenario is to have BI and analytics workloads use the read replica as the data source for reporting.

Read replicas are updated asynchronously with the native replication technology in the PostgreSQL engine. From a provisioning perspective, read replicas are considered new servers, and you manage them similarly to how you manage regular Azure Database for PostgreSQL servers. For each replica, you're billed for the provisioned compute and storage.

Using cross-region read replicas, you can scale out and balance read-heavy workloads across Azure regions, which can also be used for business continuity and disaster recovery planning. Again, you can create up to five read-only replica servers for every master server.

The online documentation for Azure Database for PostgreSQL - Single Server provides more information on read replicas, including how to create and connect to them. The article also covers how to monitor replication, stop replication, promote a replica to be the primary server, and failover to a replica, as well as other considerations when using read replication. Create and manage read replicas from the Azure portal walks you through preparing the master server, creating a read replica, stopping replication, deleting a master server, and monitoring a replica. You can also create and manage read replicas from the Azure CLI.

Logical decoding

Logical decoding in Azure Database for PostgreSQL - Single Server, another form of replication, allows you to stream data changes to external consumers. Like read replicas, logical decoding depends on the PostgreSQL write ahead log (WAL) for information. However, unlike read replication, which is built into the core PostgreSQL database engine, logical decoding uses an output plugin to convert the WAL into a readable format. Azure Database for PostgreSQL - Single Server provides the wal2json, test_decoding, and pgoutput plugins. (The pgoutput plugin is available in PostgreSQL version 10 and later versions.)

The online documentation for Azure Database for PostgreSQL - Single Server provides more information on logical decoding, including how to set up your server, start logical decoding, monitor logical decoding to prevent unconsumed logs from filling your storage, and dropping a replication slot that's not being actively consumed. This blog article covers how to use logical decoding and wal2json to capture change data from PostgreSQL and output it in JSON format.

Additional resources for Azure Database for PostgreSQL - Single Server

Following are some additional resources that can help you get started with Azure Database for PostgreSQL - Single Server:

- **Quickstarts.** Our quickstarts can help you get started with Azure Database for PostgreSQL, such as by walking you through creating a server then connecting to it and querying it. You'll find the first one here, with the rest listed immediately below it in the navigation pane.
- **Tutorials.** After trying a quickstart or two, you can follow the step-by-step tutorials to design a database, create a web app, and then monitor and tune it. Again, you'll find the first tutorial here, with the rest listed immediately below it in the navigation pane.
- **How-to guides.** Our online how-to guides can walk you through many of the things you'll likely want to do with Azure Database for PostgreSQL - Single Server. You'll find the first one here, with the rest listed immediately below it in the navigation pane.

To use the quickstarts, tutorials, and how-to guides, you'll need an Azure subscription. If you don't have an Azure subscription, you can create an Azure free account before you begin.

Finally, here are some other reference materials you may find useful:

- Limits in Azure Database for PostgreSQL - Single Server describes various types of limits in the database service—including maximum connections per pricing tier/number of vCores and functional limitations.
- Azure CLI samples for Azure Database for PostgreSQL - Single Server provides links to sample Azure CLI scripts for Azure Database for PostgreSQL - Single Server, including scripts to create a server and firewall rule, scale a server, change server configurations, restore a server, and enable and download server logs.
- Azure CLI reference for Azure Database for PostgreSQL provides a list of Azure CLI commands for managing Azure Database for PostgreSQL servers, with links to more information on each command.
- Azure Database for PostgreSQL REST API reference covers the REST operations you can use with Azure Database for PostgreSQL - Single Server. You can use these REST operations to create, delete, manage, and list servers, server configurations, databases, server logs, and firewall rules, and to list all available REST operations.

Key concepts: Flexible Server

Azure Database for PostgreSQL - Flexible Server, a fully managed database service, is designed to provide more granular control and flexibility over database management functions and configuration settings than the Single Server deployment option. It also enables greater architectural flexibility, such as collocating your database engine with the client-tier of your app to reduce latency or configuring high availability across multiple availability zones.

The Flexible Server deployment option also provides better cost optimization controls, including the ability to stop and start your server (and pay for only the time it's running) and a burstable compute tier that's ideal for workloads that don't need full, continuous compute capacity. The Flexible Server deployment option also lets you specify custom patching schedules to minimize the impact of periodic maintenance windows.

Azure Database for PostgreSQL - Flexible Server provides a deeper overview of the Flexible Server deployment option.

Supported versions

Azure Database for PostgreSQL – Flexible Server currently supports PostgreSQL major versions 13, 12, and 11. For each supported major version, new servers are created using the latest supported minor version—currently 13.3, 12.7, and 11.12. Supported versions will change over time, so it's best to check the online documentation to see which PostgreSQL versions are currently supported for Azure Database for PostgreSQL – Flexible Server.

Updates and upgrades

Existing servers are automatically upgraded to the latest minor version in your next scheduled maintenance window (described next). Currently, in-place major version upgrades—such as from version 11 to version 12—aren't supported. If you want to upgrade from one major version to a newer one, you can use PostgreSQL dump and restore or the Azure Database Migration Service to move your data to a database that was created using the newer version.

Scheduled maintenance

As a fully managed service, Azure Database for PostgreSQL - Flexible Server periodically performs scheduled maintenance on your database servers. You can let the service automatically choose a day and a time window for scheduled maintenance, in which case it will select a one-hour window between 11pm and 7am in the Azure region in which your server resides. Alternately, you can specify a maintenance window by choosing a day of the week and a one-hour period within that day. Different maintenance windows can be configured for each Flexible Server deployment in your Azure subscription.

Either way, whether you specify a maintenance window or let the service choose one for you, the service will normally alert you five days before performing any scheduled maintenance, and then let you know when scheduled maintenance is started and completed. Normally, there are at least 30 days between successful scheduled maintenance events. However, if an emergency update is needed to patch a severe vulnerability, you may not receive five days' notice, and the critical update may be applied to your server even if a successful scheduled maintenance was performed within the past 30 days.

Scheduled maintenance in Azure Database for PostgreSQL - Flexible Server provides more information on maintenance windows, including how to specify one and some things you may want to consider when doing so.

Connection libraries

Most language and client libraries used to connect to PostgreSQL server are external projects and are distributed independently. For a list of the libraries and drivers that you can use to build applications that connect to and query Azure Database for PostgreSQL - Flexible Server, see Connection libraries for Azure Database for PostgreSQL - Single Server. (This information is applicable to Flexible Server as well.)

Connection pooling

By default, PostgreSQL uses a process-based model for database connections. This makes it expensive (in terms of system resources) to maintain many idle connections and may result in resource constraints with more than a few thousand connections.

You can avoid such resource constraints by using PgBouncer, a built-in connection pooling option provided with Azure Database for PostgreSQL - Flexible Server. PgBouncer employs a more lightweight connection model, under which actual PostgreSQL connections are only used when needed—such as inside an open transaction or when a query is active. By using PgBouncer, which can be configured on a per-server basis and runs in the same VM as the PostgreSQL database server, you can support up to 10,000 database connections with low resource usage.

PgBouncer in Azure Database for PostgreSQL - Flexible Server provides more information on this topic, including how to enable and configure PgBouncer, configuring your app to use it, and considerations for using PgBouncer together with zone-redundant high availability or with other connection pools. PgBouncer is currently not supported with the Burstable server compute tier.

PostgreSQL extensions

PostgreSQL lets you extend its functionality using extensions, which bundle multiple related SQL objects together into a single package that can be loaded or removed with a command. After extensions are loaded into the database, they function like built-in features.

The PostGIS extension is a great example—it's why PostgreSQL is the world's most popular open-source geospatial database. Other popular extensions include pg_cron (for scheduling cron jobs on PostgreSQL), pg_trgm (for fuzzy string matching), xml2 (for parsing and querying xml objects), and orafce (which lets you use Oracle functions and packages within PostgreSQL). After you load them, extensions can function like built-in features.

When you deploy Azure Database for PostgreSQL - Flexible Server, the pg_stat extension is automatically installed, enabling you to track execution statistics for SQL statements. PostgreSQL extensions in Azure Database for PostgreSQL - Flexible Server provides more information on extensions, including how to load them, a list of supported extensions, and additional detail on several popular extensions.

Databases and servers

Azure Database for PostgreSQL - Flexible Server exposes access and features at the *server* level, which is created within an Azure subscription and is the parent resource for *one or more individual databases*. It collocates the computing resources for those databases in a specified Azure region, provides a namespace for the databases, and serves as a connection endpoint for server and database access. A server also defines the scope for management policies that apply to those databases—including sign-in, firewall, users, roles, and configurations.

The online documentation for Azure Database for PostgreSQL - Flexible Server provides more information on servers, including how to connect and authenticate to a database server, managing your server, and server parameters.

Compute and storage options

Under the Flexible Server deployment option, pricing for an Azure Database for PostgreSQL server is determined by the configuration of resources that you specify for that server, including the pricing tier, number of vCores, and amount of storage. Create a single database under the server, and that database has exclusive access to those resources; create multiple databases under the server, and those resources are shared.

We recently introduced a free, 12-month offer for the Flexible Server deployment option, which you can use to develop and test your applications and run small production workloads without additional costs. Available with an Azure free account, this offer provides up to 750 hours of compute time and 32GB of storage per month for the first 12 months.

Pricing tiers

You can create a server in one of three pricing tiers, which are differentiated by the amount of compute (vCores) that can be provisioned, the amount of memory per vCore, and the technology used for data storage. In general:

- **Burstable** is good for workloads that don't continuously need full compute capacity.
- **General purpose** is good for workloads that require balanced compute and memory, with scalable I/O—including web apps, mobile apps, and many other enterprise apps. This is our most popular option and works well for most scenarios.
- **Memory optimized** is good for workloads that require in-memory performance for faster transaction processing and greater concurrency—including real-time data processing and high-performance transactional or analytical apps.

After you create a server, you can adjust the number of vCores, the hardware generation, and the pricing tier up or down, within seconds. Similarly, you can increase the amount of storage or adjust your backup retention period up or down, without application downtime.

Compute and storage options in Azure Database for PostgreSQL - Flexible Server provides more information on pricing tiers, including detailed specifications for each SKU within each pricing tier. Limits in Azure Database for PostgreSQL - Flexible Server covers functional limitations for the service and associated limits for each SKU, such as the maximum number of concurrent connections.

For current pricing information, see the Azure Database for PostgreSQL pricing page. When you configure a server, the Azure portal shows the monthly cost based on the options you've selected. If you don't have an Azure subscription, you can use the Azure pricing calculator to get an estimated cost—just select Add items, expand the Databases category, choose Azure Database for PostgreSQL, then customize the options.

Reserved capacity

Azure Database for PostgreSQL - Flexible Server supports reserved capacity pricing, which lets you reserve compute power for a period of one year or three years. By taking advantage of it, you can save up to 60 percent compared to the regular, pay-as-you-go payment options.

Prepay for Azure Database for PostgreSQL - Single Server compute resources with reserved capacity provides more information on reserved capacity, including how to determine the right server size and how to purchase reserved capacity. (This article also applies to Flexible Server.) A pricing calculator can be found on the Azure Database for PostgreSQL pricing page.

Storage

When you provision a server, you'll also need to provision the amount of storage capacity available to it. This storage is used for the database files, temporary files, transaction logs, and PostgreSQL server logs. You can increase storage capacity after you create a server, but provisioned storage can't be reduced.

The total storage you provision also defines the I/O capacity available to your server in terms of I/O operations per second (IOPS). Azure Database for PostgreSQL – Flexible Server currently supports up to 16 TB of storage and up to 18,000 IOPS.

It's worth noting that IOPS are also constrained by your compute tier, including underlying VM type and number of vCores. Even though you can select any storage size independent of the server type, you may not be able to use all the IOPS that the storage can provide, especially when you choose a server with a small number of vCores. You can monitor your I/O consumption in the Azure portal or by using Azure CLI commands.

The online documentation for Azure Database for PostgreSQL - Flexible Server provides more information on provisioning storage.

Networking

When you create a server, you need to choose from one of two networking options. Your selection can't be changed after the server is created. Options include:

- **Private access (VNet integration)**, which lets you deploy your database server into an Azure Virtual Network that can be used to provide private and secure network communications.
- **Public access (allowed IP addresses)**, which lets you access your database server through a public endpoint—a publicly resolvable DNS address. With this networking option, you'll use firewall rules to define the range of IP addresses that are permitted to access your server.

The online documentation for Azure Database for PostgreSQL - Flexible Server provides more information on each networking option, including guidance on choosing one and additional detail on how each option works.

Business continuity

Azure Database for PostgreSQL - Flexible Server provides several features that you can use to help ensure business continuity and prevent data loss. Overview of business continuity with Azure Database for PostgreSQL - Flexible Server introduces these business continuity features and how you can put them to use, including an overview of recovery scenarios and how they can affect your Recovery Time Objective and Recovery Point Objective.

Backup and restore

Backups are an essential part of any business continuity strategy, providing a means to help protect data from accidental corruption or deletion. Azure Database for PostgreSQL - Flexible Server automatically backs up your server and retains those backups for between 7 and 35 days, depending on the configuration options you've selected. When you restore a backup, within the retention period, you can specify the date and time to which you want to restore your data.

All backups in Azure Database for PostgreSQL - Flexible Server are encrypted using AES 256-bit encryption and are stored in zone-redundant storage within an Azure region. The first snapshot is scheduled immediately after the server is created, after which daily snapshots are performed. Transaction logs (write ahead logs) are archived continuously, with the combination of data backups and transaction log archives enabling you to restore a server to any point-in-time within your configured backup retention period—a capability that's often-called point-in-time-restore.

Backup and restore in Azure Database for PostgreSQL - Flexible Server provides more information on how backup and restore work for Flexible Server deployments, including additional detail on backup processes, backup retention, backup storage costs, and point-in-time restore.

Zone-redundant high availability

Azure Database for PostgreSQL - Flexible Server is designed to support high availability across multiple availability zones—called zone-redundant high availability. High availability concepts in Azure Database for PostgreSQL - Flexible Server provides more information on the high availability features for Flexible Server deployments, including additional detail on steady-state operations, failover processes for planned and unplanned downtime, point-in-time restore, and zone-redundant high availability features and limitations.

Monitor and tune

Monitoring your servers not only helps you know when troubleshooting might be needed, but it also gives you the data needed for performance tuning. Azure Database for PostgreSQL - Flexible Server provides several built-in tools for monitoring and tuning, including resource usage metrics that are captured at one-minute intervals and stored for 93 days. You can also enable logging on your server, and then send those logs to Azure Monitor, Event Hubs, or an Azure Storage account. In addition, the Query Store feature in Azure Database for PostgreSQL - Flexible Server lets you track query performance over time to help you find long-running and resource-intensive queries.

The online documentation for Azure Database for PostgreSQL - Flexible Server provides more information on its monitoring and tuning features.

Logging and audit logs

Azure Database for PostgreSQL - Flexible Server lets you configure and access standard PostgreSQL logs, which can be used to help identify, troubleshoot, and repair configuration errors and performance issues. Types of data you can log include errors, query information, auto-vacuum records, connections, and checkpoints.

Logs in Azure Database for PostgreSQL - Flexible Server covers how to configure logging on your server, access log files, and configure diagnostic settings to send your PostgreSQL logs to Azure Monitor, Event Hubs, and Azure Storage. Audit logging in Azure Database for PostgreSQL - Flexible Server covers how to configure detailed session and object audit logging, as enabled by the pgAudit extension for PostgreSQL.

Understanding query performance using Query Store

The Query Store feature in Azure Database for PostgreSQL - Flexible Server provides a way to track query performance over time. It simplifies performance-troubleshooting by helping you quickly find the longest running and most resource-intensive queries, which it does by automatically capturing a history of queries and runtime statistics, retaining those results for your review, and slicing them by time so that you can identify temporal usage patterns.

The online documentation for Azure Database for PostgreSQL - Flexible Server provides more information on Query Store, including usage scenarios and best practices.

Azure Advisor recommendations

The Azure Advisor system uses telemetry to issue performance and reliability recommendations for Azure Database for PostgreSQL - Flexible Server, which are made available in the Azure portal. In general, there are three types of recommendations:

- Performance recommendations – based on CPU usage, memory pressure connection pooling, disk utilization, and other server parameters.
- Reliability recommendations – based on storage limits and connection limits.
- Cost optimization – including recommendations on server right-sizing.

When recommendations are available, a preview will appear as a banner notification in the Azure portal, under Overview. Details can be viewed in the Notifications section, just below the resource usage graphs. The online documentation provides more information on Azure Advisor for Azure Database for PostgreSQL.

Logical replication and logical decoding

Azure Database for PostgreSQL - Flexible Server supports both logical replication and logical decoding. They both allow you to replicate data out of PostgreSQL, using the database engine's write-ahead log (WAL) as the source of changes. At a high level, the differences between logical replication and logical decoding include the following:

- Logical replication allows you to specify a table or set of tables to be replicated between PostgreSQL instances.
- Logical decoding extracts changes across all tables in a database and *cannot* directly send data between PostgreSQL instances.

Logical replication and logical decoding in Azure Database for PostgreSQL - Flexible Server provides additional information on these two data extraction and replication methodologies.

Security

Many of the security features for Azure Database for PostgreSQL - Flexible Server are currently in development. Our goal is to at least achieve feature-parity with the security features in the Single Server deployment option. As new security features are added to Flexible Server, we'll update the online documentation for Azure Database for PostgreSQL, post about it in Azure updates, and may write about it on the Azure Database for PostgreSQL blog.

Additional resources for Azure Database for PostgreSQL - Flexible Server

Following are some additional resources that can help you get started with Azure Database for PostgreSQL - Flexible Server:

- **Quickstarts.** Our quickstarts can help you get started with Azure Database for PostgreSQL - Flexible Server, such as creating a server then connecting to it and querying it. You'll find the first one here, with the rest listed immediately below it in the navigation pane.

- **Tutorials.** After trying a quickstart or two, you can follow the step-by-step tutorials to design a database, create a web app, and then monitor and tune it. Again, you'll find the first tutorial here, with the rest listed immediately below it in the navigation pane.
- **How-to guides.** Our online how-to guides can walk you through many of the things you'll likely want to do with Azure Database for PostgreSQL - Flexible Server. You'll find the first one here, with the rest listed immediately below it in the navigation pane.

To use the quickstarts, tutorials, and how-to guides, you'll need an Azure subscription. If you don't have an Azure subscription, you can create an Azure free account before you begin.

Finally, here are some other reference materials you may find useful:

- Limits in Azure Database for PostgreSQL - Flexible Server describes various types of limits in the database service—including maximum connections per pricing tier/number of vCores and functional limitations.
- Azure CLI reference for Azure Database for PostgreSQL provides a list of Azure CLI commands for managing Azure Database for PostgreSQL servers, with links to more information on each command.
- Azure Database for PostgreSQL REST API reference covers the REST operations you can use with Azure Database for PostgreSQL - Flexible Server. You can use these REST operations to create, delete, manage, and list servers, server configurations, databases, server logs, and firewall rules, and to list all available REST operations.

Key concepts: Hyperscale (Citus)

The Hyperscale (Citus) deployment option for Azure Database for PostgreSQL horizontally scales across multiple servers using sharding, with its query engine parallelizing incoming SQL queries across these servers for faster responses on large datasets. Applications built for PostgreSQL can run distributed queries on Azure Database for PostgreSQL - Hyperscale (Citus) with standard connection libraries and minimal changes.

The Hyperscale (Citus) deployment option is a good choice for applications that require greater scalability and performance than the Single Server or Flexible Server deployment options can provide—a good rule of thumb being workloads that are approaching or already exceed 100GB of data.

Supported versions

Azure Database for PostgreSQL - Hyperscale (Citus) currently supports PostgreSQL versions 11, 12, and 13. Supported database versions in Azure Database for PostgreSQL – Hyperscale (Citus) provides more information on supported PostgreSQL versions for Hyperscale (Citus) deployments, including the latest minor version.

Each major version includes the latest validated Citus extension for that major version. For example, currently, PostgreSQL versions 11 and 12 include Citus 9.5-1, whereas PostgreSQL version 13 comes with Citus 10.0-2. If you an in-place upgrade to a new major PostgreSQL version (discussed next), the latest validated Citus extension for that major version is also installed.

Updates and upgrades

Existing servers in a Hyperscale (Citus) server group are automatically upgraded to the latest supported minor version in your next scheduled maintenance window (described next). Upgrade a Hyperscale (Citus) server group describes how to upgrade to a new major version of PostgreSQL on all server group nodes. (Microsoft recommends that you test the upgrade first, as noted in the article.)

Scheduled maintenance

As a fully managed service, Azure Database for PostgreSQL – Hyperscale (Citus) periodically performs scheduled maintenance on your database servers. You can let the service automatically choose a day and a time window for scheduled maintenance, in which case it will select a one-hour window between 11pm and 7am in the Azure region in which your server group resides. Alternately, you can specify a maintenance window by choosing a day of the week and a one-hour period within that day. Different maintenance windows can be configured for each Hyperscale (Citus) deployment in your Azure subscription.

Either way, whether you specify a maintenance window or let the service choose one for you, the service will normally alert you five days before performing any scheduled maintenance, and then let you know when scheduled maintenance is started and completed. Normally, there are at least 30 days between successful scheduled maintenance events. However, if an emergency update is needed to patch a severe vulnerability, you may not receive five days' notice, and the critical update may be applied to your server even if a successful scheduled maintenance was performed within the past 30 days.

Scheduled maintenance in Azure Database for PostgreSQL – Hyperscale (Citus) provides more information on maintenance windows, including how to specify one and some things you may want to consider when doing so.

Connection pooling

By default, PostgreSQL uses a process-based model for database connections. This makes it expensive (in terms of system resources) to maintain many idle connections and may result in resource constraints with more than a few thousand connections.

You can avoid such resource constraints by using PgBouncer, a built-in connection pooling option provided with Azure Database for PostgreSQL - Hyperscale (Citus). PgBouncer employs a more lightweight connection model, under which actual PostgreSQL connections are only used when needed—such as inside an open transaction or when a query is active. By using PgBouncer, you can support up to 2,000 simultaneous connections to the coordinator node of your server group.

PostgreSQL extensions

PostgreSQL lets you extend its functionality using extensions, which bundle multiple related SQL objects together into a single package that can be loaded or removed with a command. After extensions are loaded into the database, they function like built-in features. Popular extensions include PostGIS (for geospatial data), pg_cron (for scheduling cron jobs on PostgreSQL), pg_trgm (for fuzzy string matching), xml2 (for parsing and querying xml objects), and orafce (which lets you use Oracle functions and packages within PostgreSQL).

When Azure Database for PostgreSQL - Hyperscale (Citus) is deployed, the Citus and pg_stat_statements extensions are automatically installed. PostgreSQL extensions in Azure Database for PostgreSQL – Hyperscale (Citus) provides a list of other supported extensions, including how to install them.

Distributed data

To optimize performance for Hyperscale (Citus) deployments, you'll need to understand how it employs sharding to distribute rows of data across worker nodes. After you understand the basics, you'll be able to determine the proper distribution columns based on your application type and ensure data is collocated for optimal query performance.

Nodes and tables

Under the Hyperscale (Citus) deployment option, individual PostgreSQL servers (called nodes) in a server group coordinate with one another in a "shared nothing" architecture. The nodes in a server group collectively hold more data and provide more CPU cores than possible on a single server, and the server group is easily scaled by simply adding more nodes.

With the exception of the Basic Tier for Hyperscale (Citus) deployments (discussed below), every server group has a coordinator node and multiple worker nodes. Applications send queries to the coordinator, which relays it to the relevant workers and combines the returned results. Applications cannot connect directly to worker nodes.

Under this architecture, distributed tables are the key to achieving scalable performance. Such distribution is achieved via sharding—that is, by storing the different rows in a table on different worker nodes. Database administrators must specify how data is sharded; failure to distribute tables leaves them entirely on the worker node, preventing the use of cross-server parallelism.

For each query, the coordinator node either routes it to a single worker node or parallelizes it across several workers, depending on where the required data resides. The coordinator determines how to distribute queries by consulting metadata tables that track data distribution across worker nodes, along with their DNS names and health.

The online documentation for Azure Database for PostgreSQL - Hyperscale (Citus) provides more information on coordinator and worker nodes, table types, and how distributed tables are stored as shards on worker nodes.

Determining your application type

Running efficient queries on a Hyperscale (Citus) server group requires that tables be properly distributed across worker nodes. To optimize data distribution, you'll first need to model your data, which means considering the type of application you have and its query patterns.

From a broad perspective, there are two types of applications—multi-tenant and real-time—that work especially well on Hyperscale (Citus) deployments. The first step in modeling your data is to identify which of them more closely resembles your application, which you can do by using the guidance on application types provided in the online documentation.

Choosing a distribution column

Azure Database for PostgreSQL - Hyperscale (Citus) maps rows of data to specific shards based on the value of each row's distribution column. Because of this, in modeling your data, choosing a distribution column for each table is one of the most important design decisions you'll make.

Choosing a good distribution column groups related data together on the same physical nodes, which ensures fast queries and support for all SQL features. Similarly, choosing a bad distribution column can reduce query performance and prevent SQL features from working across nodes.

Choose distribution columns in Azure Database for PostgreSQL – Hyperscale (Citus) provides guidance on choosing distribution columns for multi-tenant and real-time apps, the two most common types of Hyperscale (Citus) deployments.

Table colocation

Data colocation—storing related data together on the same worker nodes—can minimize the need for network traffic between nodes, leading to better parallelism and ultimately better query performance. Table colocation in Azure Database for PostgreSQL – Hyperscale (Citus) discusses data colocation in greater detail and provides a practical example based on a multi-tenant app.

Columnar storage

Azure Database for PostgreSQL - Hyperscale (Citus) supports append-only columnar table storage for analytic and data warehousing workloads. When columns (rather than rows) are stored contiguously on disk, data becomes more compressible and queries can request a subset of columns more quickly. Columnar table storage discusses how to use this feature, including how to measure compression, examples, limitations, and other considerations.

Configuration options and pricing

You can configure compute and storage resources independently for coordinator and worker nodes in an Azure Database for PostgreSQL - Hyperscale (Citus) server group. Compute resources are provisioned as vCores, representing the logical CPUs of the underlying hardware. Provisioned storage is the total storage capacity available to the coordinator and worker nodes in your server group—used for database files, temporary files, transaction logs, and PostgreSQL server logs.

Azure Database for PostgreSQL – Hyperscale (Citus) configuration options provides more detail on compute and storage options for Hyperscale (Citus) deployments, along with a list of the Azure regions in which the service is available.

Basic tier

The basic tier in Azure Database for PostgreSQL - Hyperscale (Citus) is a simple way to create a small server group that you can scale out later—and get started with lower costs. While server groups in the standard tier have a coordinator node and at least two worker nodes, the basic tier runs everything in a single database node. The online documentation provides more information on the basic tier for Azure Database for PostgreSQL - Hyperscale (Citus).

Pricing

For current pricing information for Hyperscale (Citus) deployments, see the Azure Database for PostgreSQL pricing page. When you configure a Hyperscale (Citus) deployment, the Azure portal shows the monthly cost on the Compute + Storage tab based on the options you've selected. If you don't have an Azure subscription, you can use the Azure pricing calculator to get an estimated price—just expand the Databases category at the left, select Azure Database for PostgreSQL, then click the View link from the flyout. After the new page loads, select Hyperscale (Citus) as the deployment option (it's at the top), and then customize the other options provided.

Reserved capacity

Azure Database for PostgreSQL – Hyperscale (Citus) supports reserved capacity pricing, which lets you reserve compute power for a period of one year or three years. By taking advantage of it, you can save up to 65 percent compared to the regular, pay-as-you-go payment options.

Prepay for Azure Database for PostgreSQL - Hyperscale (Citus) compute resources with reserved capacity provides more information on reserved capacity, including how to determine the right server group size, how to purchase reserved capacity, and other considerations. A pricing calculator can be found on the Azure Database for PostgreSQL pricing page.

Limits and limitations

Azure Database for PostgreSQL – Hyperscale (Citus) limits and limitations describes the functional and capacity limits for Hyperscale (Citus) deployments, such as maximum storage size and the maximum number of connections to coordinator and worker nodes.

Security

To enable applications to connect to your database—specifically, to its coordinator node—you'll need to configure firewall rules on the server and configure your apps to use SSL. Beyond the use of these mandatory security mechanisms, you can use the Azure security baseline for Azure Database for PostgreSQL - Hyperscale (Citus) to help maximize security.

Firewall rules

By default, the server firewall for Azure Database for Azure Database for PostgreSQL - Hyperscale (Citus) prevents all access to your coordinator node until you specify which computers have permission. Called firewall rules, such permissions can be configured by using the Azure portal and are based on one or more ranges of specified IP addresses. To create server-level firewall rules, you must be the subscription owner or a subscription contributor.

Firewall rules in Azure Database for PostgreSQL - Hyperscale (Citus) provides more information on firewall rules, including an overview of how they work and tips for troubleshooting. There's also a how-to guide on configuring firewall rules by using the Azure portal.

Configure TLS

With Hyperscale (Citus) deployments, the coordinator node requires client applications to connect with Transport Layer Security (TLS) to help protect data in transit. Configure TLS in Azure Database for PostgreSQL - Hyperscale (Citus) discusses considerations for various connectors and application frameworks and the use of certificate verification for TLS connectivity.

Security baseline

The Azure Security Benchmark contains recommendations to help you improve the security of your applications and data on Azure. It focuses on cloud-centric control areas. These controls are consistent with well-known security benchmarks, such as those described by the Center for Internet Security (CIS) Controls Version 7.1.

Azure security baseline for Azure Database for PostgreSQL – Hyperscale (Citus) applies guidance from the Azure Security Benchmark version 1.0 to Hyperscale (Citus) deployments. Topics are grouped by the security controls defined by the Azure Security Benchmark, including network security, logging and monitoring, identity and access control, data protection, vulnerability management, inventory and asset management, secure configuration, malware defense, data recovery, incident response, penetration tests, and red team exercises—with links to the Azure Security Benchmark documentation for more information on each topic.

Business continuity

Azure Database for PostgreSQL – Hyperscale (Citus) provides several features that you can use to help ensure business continuity and prevent data loss, including automated backups and built-in high availability.

Backup and restore

Backup and restore are an essential part of any business continuity strategy, providing a means to help protect against accidental corruption or deletion. Azure Database for PostgreSQL – Hyperscale (Citus) automatically creates backups of each node in a server group and stores them in locally redundant storage, enabling you to restore your Hyperscale (Citus) server group to a specified time within the service's 35-day backup retention period.

All backups are encrypted using AES 256-bit encryption. In Azure regions that support availability zones, backup snapshots are stored in three availability zones. As long as at least one availability zone is online, the Hyperscale (Citus) server group can be restored.

The online documentation provides more information on backup and restore in Azure Database for PostgreSQL - Hyperscale (Citus).

High availability

Azure Database for PostgreSQL - Hyperscale (Citus) helps avoid database downtime by maintaining standby replicas of every node in a server group. If a node goes down, the service switches incoming connections from the failed node to its standby. To properly take advantage of the high-availability mechanisms built into Azure Database for PostgreSQL - Hyperscale (Citus), applications need to detect and retry dropped connections and failed transactions when accessing the coordinator node.

High availability in Azure Database for PostgreSQL – Hyperscale (Citus) provides more detail about failover and recovery for Hyperscale (Citus) deployments.

Monitor and tune

Monitoring your servers not only helps you know when troubleshooting might be needed, but it also gives you the data needed for performance tuning. Azure Database for PostgreSQL – Hyperscale (Citus) provides several built-in tools for monitoring and tuning, including resource usage metrics for each node in a server group. These metrics are captured at one-minute intervals and stored for up to 30 days.

Monitor and tune Azure Database for PostgreSQL - Hyperscale (Citus) provides a detailed list of the metrics that are captured. You can set up alerts on these metrics by using the Azure portal.

Audit logs

Audit logging of database activities in Azure Database for PostgreSQL - Hyperscale (Citus) is available through the PostgreSQL pgAudit extension. By default, pgAudit log statements are emitted along with your regular log statements by using the standard logging facility in PostgreSQL.

Audit logging in Azure Database for PostgreSQL - Hyperscale (Citus) provides more information on audit logging, including usage considerations, enabling pgAudit, pgAudit settings, audit log format, getting started with audit logs, and viewing audit logs. There's also a how-to article on sending audit logs to a specified destination.

Replication

The read replica feature in Azure Database for PostgreSQL - Hyperscale (Citus) allows you to replicate data from a Hyperscale (Citus) server group to a read-only server group. Replicas are updated asynchronously using PostgreSQL physical replication, and you can replicate from the primary server group to an unlimited number of replica server groups. Read replicas in Azure Database for PostgreSQL - Hyperscale (Citus) provides more information on read replication for Hyperscale (Citus) deployments, including when you might want to use it, creating and connecting to a replica, and other considerations.

Azure Arc: Running Azure Database for PostgreSQL on any infrastructure

Azure Arc extends the capabilities of Azure Resource Manager to any infrastructure, including across data centers, the edge, and multicloud environments. You can deploy Azure Database for PostgreSQL - Hyperscale (Citus) as an Azure Arc-enabled data service, which uses Kubernetes to let you run Azure Database for PostgreSQL - Hyperscale (Citus) on the infrastructure of your choice.

Deploying Azure Database for PostgreSQL - Hyperscale (Citus) via Azure Arc provides you with access to all the benefits of Azure—including pay-as-you-go pricing, elastic scalability, and unified management—for workloads with or without a direct connection to the cloud. It can also help you meet compliance and governance requirements, such as those associated with data sovereignty or other regulatory mandates.

Additional resources for Azure Database for PostgreSQL – Hyperscale (Citus)

Following are some additional resources that can help you get started with Azure Database for PostgreSQL - Hyperscale (Citus):

- **Quickstarts.** Our quickstarts can help you get started with Azure Database for PostgreSQL - Hyperscale (Citus), such as creating a server then connecting to it and querying it. You'll find the first one here, with the rest listed immediately below it in the navigation pane.
- **Tutorials.** After trying a quickstart or two, you can follow the step-by-step tutorials to design a database, create a web app, and then monitor and tune it. Again, you'll find the first tutorial here, with the rest listed immediately below it in the navigation pane.

- **How-to guides.** Our online how-to guides can walk you through many of the things you'll likely want to do with Azure Database for PostgreSQL - Hyperscale (Citus). You'll find the first one here, with the rest listed immediately below it in the navigation pane.

To use the quickstarts, tutorials, and how-to guides, you'll need an Azure subscription. If you don't have an Azure subscription, you can create an Azure free account before you begin.

Finally, here are some other reference materials you may find useful:

- Azure Database for PostgreSQL - Hyperscale (Citus) limits and limitations describes various types of limits in the database service—including maximum connections and functional limitations.
- Functions in the Hyperscale (Citus) SQL API provides reference information for the user-defined functions that enable horizontal scalability Hyperscale (Citus) deployments.
- Server parameters covers the various parameters that affect the behavior of Hyperscale (Citus) deployments, both from standard PostgreSQL and specific to Hyperscale (Citus).
- System tables and views discusses the special tables that Azure Database for PostgreSQL - Hyperscale (Citus) creates and maintains to manage metadata for distributed tables.

Use cases

At a high level, the usage scenarios for Azure Database for PostgreSQL can be divided into two types: migrations from another database, and new cloud-native apps. Migrations to Azure Database for PostgreSQL are typically from other PostgreSQL databases or from Oracle. Many cloud-native apps built on Azure Database for PostgreSQL fall into one or more of the following four categories: transactional, geospatial-aware, SaaS, and real-time.

Migrations

Homogeneous Migration

Seamless migration with support for latest open-source database editions

Fast and secure migration with minimal downtime

Oracle to Postgres Migration

Complete migrations faster and save money while enabling hyperscale

Cloud-Native Applications

Transactional applications

Engage cloud native developers with the flexibility of AKS and the extensibility of PostgreSQL

Geospatial-aware applications

Leverage PostGIS, the world's most popular OS geospatial solution, to deliver location-aware experiences

SaaS applications

Enjoy high-throughput OLTP and sharding for multi-tenant apps with Hyperscale (Citus)

Real-time applications

Ensure real-time data with support for time-series data and JSONB

Combine relational data with semi-structured data types

Migrations from other databases

Customers often migrate to Azure Database for PostgreSQL from other PostgreSQL databases, or from Oracle. Azure Database Migration Guides provide step-by-step guidance on migrating to various Azure data services, in addition to a selection of case studies and partner tools.

Homogeneous migration (PostgreSQL to Azure Database for PostgreSQL)

Azure Database for PostgreSQL is a frequent target for migrations from an existing PostgreSQL database, which might be running on-premises, in an IaaS environment on a hosted VM, or on another cloud provider's managed PostgreSQL service, such as Amazon RDS. Regardless of where your PostgreSQL database runs today, by migrating to Azure Database for PostgreSQL, a fully managed service, you'll be able to quickly modernize your apps and shift your focus from database management to app development. You'll also benefit from built-in high availability—including a financially backed SLA of 99.99%—and advanced security and compliance controls.

Migrations from PostgreSQL to Azure Database for PostgreSQL are often done using the Azure Database Migration Service, making the process fast and seamless with minimal downtime. There's also an on-premises PostgreSQL to Azure Database for PostgreSQL Migration Guide.

HarvestMark is an example of a customer who has benefited from migrating its PostgreSQL databases to Azure Database for PostgreSQL.

Migration from Oracle

Azure Database for PostgreSQL is also a frequent target for migrations off of Oracle, enabling companies to reduce TCO while reaping the benefits of cloud-scale, built-in intelligence, and deep integration with the rest of the Azure ecosystem. It's based on the broadly embraced, open-source PostgreSQL project, not a forked version, eliminating any chance of vendor lock-in.

Similarities between Oracle and PostgreSQL make such migrations easier than moving from Oracle to many other relational databases, and you can leverage free open-source tools like Ora2PG to automate many schema changes. Migrate Oracle to Azure Database for PostgreSQL provides a closer look at this process. There's also an Oracle to Azure Database for PostgreSQL Migration Guide.

Cloud-native apps

Azure Database for PostgreSQL is often chosen as the data tier for new cloud-native apps. Built from the ground up to take advantage of cloud scale and performance, such apps are typically based on a microservices architecture, make extensive use of fully managed services, and take advantage of continuous delivery to improve reliability and accelerate time to market. Regardless of what the app does, building it on Azure Database for PostgreSQL means you'll benefit from cloud scale, built-in intelligence, advanced security and compliance features, built-in high availability, and integration with the broader Azure ecosystem.

Transactional apps

Azure Database for PostgreSQL is often chosen for transactional apps because of its robust support for many data types, including time-series and JSONB. PostgreSQL transactions are ACID (Atomicity, Consistency, Isolation, and Durability) complaint, ensuring data consistency. The Hyperscale (Citus) deployment option is particularly well-suited for high-throughput OLTP scenarios, with the ability to support tens of thousands of transactions per second.

Customers who have benefited from building transactional apps on Azure Database for PostgreSQL include Finxact, Swiss Re, and Robert Bosch GmbH.

Geospatial-aware apps

Azure Database for PostgreSQL supports the highly popular PostGIS extension for PostgreSQL, making it a strong choice for apps with a location component—such as route optimization, fleet monitoring, or supply chain management. Again, the Hyperscale (Citus) deployment option can be used to support geospatial apps that require higher levels of database throughput.

Customers who have benefited from building geospatial-aware apps on Azure Database for PostgreSQL include Helsinki Region Transport Authority and Cloud9.

SaaS apps

JSONB support in PostgreSQL makes Azure Database for PostgreSQL a strong candidate for SaaS apps, enabling you to combine relational data with semi-structured data types. The Hyperscale (Citus) deployment option is especially well suited for SaaS apps; they're typically multitenant, and tenant ID is a natural distribution key for sharding. With Azure Database for PostgreSQL - Hyperscale (Citus), you can keep your tenants separate while retaining the ability to run cross-tenant queries for internal monitoring.

The Hyperscale (Citus) deployment option is also a good choice for apps that require high concurrency—for example, those that must concurrently ingest data while supporting dozens or hundreds of concurrent end-user queries. Similarly, it's a good choice for fast-growing SaaS apps that need to be able to scale to support thousands of customers, hundreds of thousands of users, and hundreds of gigabytes of data.

We often see customers using Azure Database for PostgreSQL to build highly scalable sales and marketing automation apps, which often:

- Take advantage of the extensibility and JSON support provided by PostgreSQL to deliver rich functionality.
- Employ a multitenant architecture with one Azure Database for PostgreSQL server per customer.
- Use read replicas to improve performance—achieved by offloading read-intensive workloads (such as reporting) from the primary read-write replica.
- Rely on Azure Database for PostgreSQL for the reliability, high availability, and performance needed to meet customer expectations—for example, in the retail industry, the ability to easily scale during the holiday season and other peak shopping periods.

Real-time apps

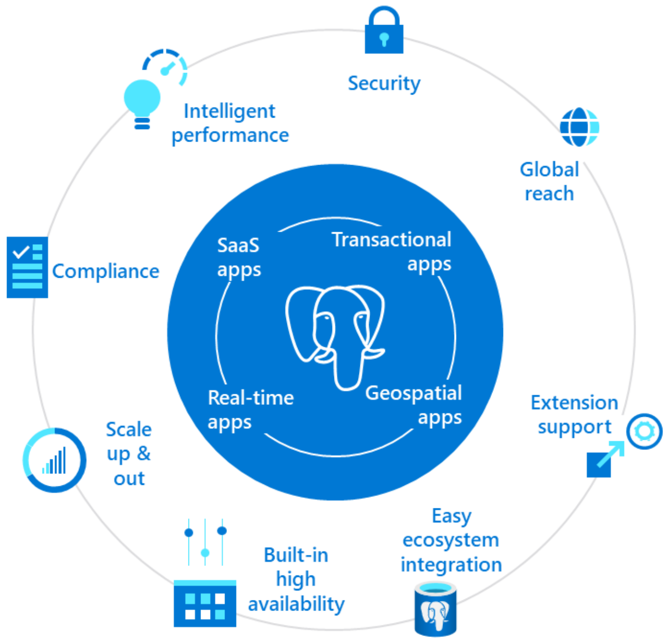
Support for time-series and JSONB makes Azure Database for PostgreSQL a good fit for real-time apps, letting you combine relational data with semi-structured data. The Hyperscale (Citus) deployment option can be used for high-throughput scenarios, leveraging parallelism to achieve sub-second query speeds. Hyperscale (Citus) is also good for high-concurrency or mixed workloads that combine transactions and analytics, with its sharding mechanisms well-suited to meeting the needs of users who need to ask questions with many parameters—implemented as queries with many different column filters.

We often see customers using Azure Database for PostgreSQL to build customer-facing dashboards capable of supporting low-latency, interactive queries. A common architectural pattern is to use Azure Databricks or Azure Data Factory to clean and transform the data before storing it Azure Database for PostgreSQL – Hyperscale (Citus). JSONB support enables the storage of semi-structured data with 6-7x compression rates, with final datasets exposed for end-user access via interactive, real-time analytics tools such as Metabase or Power BI. The Hyperscale (Citus) deployment option also provides the ability to scale horizontally without downtime, and to optimize costs by independently scaling each node's compute and storage. This blog article describes the journey of one such customer, who migrated its interactive analytics environment from Amazon Redshift to Azure Database for PostgreSQL – Hyperscale (Citus).

IoT apps are another type of real-time app that customers are building with Azure Database for PostgreSQL – Hyperscale (Citus). Its horizontal scalability enables mixed-workload (OLAP + OLTP) solutions capable of ingesting and processing telemetry from millions of devices, with the ability to simultaneously power event-driven apps and dashboards. Device ID is often used as the sharding key for such solutions.

We've seen customers use Azure Database for PostgreSQL – Hyperscale (Citus) to build IoT solutions that capture up to tens of terabytes of data from millions of devices. One test scenario simulated 2 billion measurements per hour ingested from 5 million devices, with a P95 (query performance in 95 percent of cases) of less than 500ms for non-hierarchical queries and less than two seconds for hierarchical queries.

BNY Mellon is an example of a customer who has benefited from building real-time apps on Azure Database for PostgreSQL.



Building applications on Azure

In the preceding pages, we've covered why Azure Database for PostgreSQL is a great database service. However, rarely does a database exist in isolation. More likely than not, it's purpose is to function as a data store for one or more apps, which means those apps need to work well with it, efficiently storing and retrieving data. Similarly, those apps need to be developed, deployed, monitored, and scaled alongside your database. If all the parts of your app don't work well together throughout the application lifecycle, you're leaving a lot of the benefits of fully managed PaaS services off-the-table.

So what does it take to achieve this? At a minimum, it requires a fully managed application hosting environment that's as rich and robust as your database—albeit in slightly different ways. For development, you'll need to be able to use the same familiar tools, with support for your desired development languages and frameworks. For deployment, you'll want to integrate changes to your database into your CI/CD pipelines, test and staging environments, and more.

In production, you'll want your app collocated with your database to optimize performance. You'll want to monitor your app hosting and development environments using an integrated set of tools, knowing which tier needs to scale and why. Finally, when scalability is needed, you'll want to be able to scale your application hosting environment as easily as you can scale your database, using the same familiar mechanisms. All of these benefits are reasons why you should consider hosting your app on Azure as well.

Choosing an app hosting environment

First, let's examine some of the most popular application hosting environments on Azure and what they can bring to the table alongside Azure Database for PostgreSQL.

Azure App Service

Customers often use Azure Database for PostgreSQL together with Azure App Service, a fully managed HTTP-based service for building and hosting web apps, mobile back ends, and RESTful APIs—all without having to manage infrastructure. You can work in the programming language of your choice, including .NET, .NET Core, Java, Ruby, Node.js, PHP, and Python. App Service provides built-in auto-scaling and high availability, supports both Windows and Linux, and enables automated deployments from GitHub, Azure DevOps, or any Git repo.

With App Service, you pay only for the Azure compute resources you use, which are determined by the App Service plan that you choose. For apps that require superior levels of scalability, you can run in a fully dedicated and isolated App Service Environment. Along with Azure Spring Cloud (discussed next), App Service is a good choice for companies that want to migrate their on-premises databases off of Oracle and need a suitable application hosting environment.

If you'd like to try out using App Service and Azure Database for PostgreSQL for yourself, Deploy a Django app on App Service with Azure Database for PostgreSQL - Flexible Server provides step-by-step instructions for doing so.

Azure Spring Cloud

Azure Spring Cloud makes it easy to deploy Java Spring Boot and ASP.NET Steeltoe apps to Azure without any code changes. The Spring Cloud service manages infrastructure so that you can focus on your code, with lifecycle management features that include monitoring and diagnostics, configuration management, service discovery, CI/CD integration, and blue-green deployments.

You can migrate existing Spring apps to Spring Cloud to manage cloud scaling and costs, or you can modernize apps with Spring Cloud patterns to improve agility and speed of delivery. Either way, Spring Cloud is a great way to run Java at cloud scale, without complicated infrastructure. You'll also benefit from rapid development and deployment, without containerization dependencies, along with efficient and effortless monitoring of your production workloads.

Azure Kubernetes Service

Cloud-native apps that use Azure Database for PostgreSQL are often deployed on Azure Kubernetes Service (AKS), a fully managed Kubernetes environment, which provides a great way to unite your development and operations teams. AKS supports integrated CI/CD along with enterprise-grade security and governance, providing a great way to rapidly build, deliver, and scale your apps with confidence.

AKS achieves this by offloading the complexity and operational overhead of managing a Kubernetes cluster to Azure, handling critical tasks like health monitoring and maintenance. Upon cluster deployment, the Kubernetes master and all nodes are deployed and configured for you. Advanced networking, Azure Active Directory (Azure AD) integration, monitoring, and other features also can be configured during the deployment process. You only manage and maintain (and pay for) the agent nodes within your cluster, not the masters, which are handled by Azure.

If you'd like to try out using AKS and Azure Database for PostgreSQL for yourself, Deploy a Django app on AKS with Azure Database for PostgreSQL - Flexible Server provides step-by-step instructions for doing so. Connecting Azure Kubernetes Service and Azure Database for PostgreSQL covers some options to consider when using AKS with Azure Database for PostgreSQL, including accelerated networking, Open Service Broker for Azure, and connection pooling.

Designing an application architecture

Before building your app, you'll want to review our guidance for architecting solutions on Azure using established patterns and best practices. The Azure Architecture Center provides a wealth of information in this area—from designing for the cloud to choosing the right technology and optimizing for your workload.

Within the Azure Architecture Center, you'll find the Azure Application Architecture Guide, which presents a structured approach for designing applications on Azure that are scalable, secure, resilient, and highly available. It's based on proven practices that Microsoft has learned from customer engagements.

The Azure Architecture Center is also full of ideas for solutions that use Azure Database for PostgreSQL, including finance management, intelligent apps, and retail and e-commerce.

Common app development best practices

Regardless of the type of app you're building, you'll want to use established best practices for connecting your app to its database. Best practices for building an application with Azure Database for PostgreSQL discusses configuration of application and database resources, things you can do to boost performance and resiliency, database deployment options, and considerations to keep in mind when building your database schema and queries.

Deploying Azure Database for PostgreSQL with GitHub actions

GitHub provides many benefits for application developers, including comprehensive source control and the ability to track changes across multiple versions of the code base. Database schemas can evolve similarly, with multiple versions across development, test, and production environments—and thus require similar robust source control capabilities.

GitHub Actions for Azure can be used to automate database development workflows within GitHub, including deployment of database updates to Azure Database for PostgreSQL. Use GitHub Actions to connect to Azure Database for PostgreSQL provides an overview of how it works, beginning with the workflow file that defines the various steps and parameters. The article also covers how to generate deployment credentials, specify a connection string, configure GitHub secrets, add your workflow, and review your deployment. You can find a sample project on GitHub here.

Conclusion

Microsoft Azure provides an end-to-end platform for open-source applications, including a fully managed PostgreSQL database service that automatically handles all maintenance, patching, and updates to let you focus on application innovation. Azure Database for PostgreSQL can be provisioned in minutes and scaled in seconds, with a simple yet flexible pricing model that includes automatic patching and backups, monitoring tools, essential security features, and more. And because it's a part of Azure, when you choose Azure Database for PostgreSQL, you'll have everything you need to build great apps and deploy them across the globe.

With Azure Database for PostgreSQL, you'll get support for the latest community versions of PostgreSQL, a simplified developer experience, and AI-powered performance optimization. You'll also get a 99.99-percent service level agreement, advanced data security and compliance features, and multiple pricing options—including the ability to pay-as-you-go or to save money with Reserved Capacity.

The new Flexible Server deployment option provides even more, including custom maintenance windows, more configuration parameters, zone-redundant high availability, and additional cost optimization controls. If performance is paramount, the Hyperscale (Citus) deployment option delivers virtually limitless scalability and performance, including a basic pricing tier that makes it easy and inexpensive to get started. Hyperscale (Citus) server groups can even be deployed as an Azure Arc-enabled data service, letting you run Azure Database for PostgreSQL outside of Azure on the infrastructure of your choice.

Sign up for an Azure free account and get started with Azure Database for PostgreSQL today. If you already have an Azure account, you can create a database on the Azure portal.

Additional resources

If you'd like to learn more about Azure Database for PostgreSQL, here are some resources:

- Online documentation for Azure Database for PostgreSQL
- Pricing information for Azure Database for PostgreSQL
- Azure pricing calculator for Azure Database for PostgreSQL
- Azure Database for PostgreSQL blog
- Microsoft Learn for online training courses
- Tech Community discussion forums for Azure Database for PostgreSQL
- Azure Tips and Tricks for helpful tips for developers
- Azure service health dashboard
- Azure product availability by region
- Azure Community Support
- Azure Blog and Updates