# PostgreSQL database migration guide to Azure

## Latest PaaS and IaaS deployments

This project was commissioned by Microsoft and AMD.

# Table of contents


Introduction 1

PaaS and IaaS: What to know for PostgreSQL deployments 1

PaaS Deployments: Azure Database for PostgreSQL - Flexible Server 2

Best practices for deploying Azure Database for PostgreSQL – Flexible Server 2

Understanding Azure Database for PostgreSQL – Flexible Server 2

Benefits of deploying Azure Database for PostgreSQL – Flexible Server 3

Planning for deployment 4

Azure compute overview 4

Azure VM service tiers and workload considerations 4

AMD processor-based VM series to consider 5

Configurable options and limitations 5

Storage considerations 6

Backup considerations 6

Deploying Azure Database for PostgreSQL – Flexible Server 7

Creating an Azure Database for PostgreSQL – Flexible Server via the Azure portal 7

Creating an Azure Database for PostgreSQL – Flexible Server via Azure CLI 12

Overview of using Azure Resource Manager templates 13

Managing Azure Database for PostgreSQL – Flexible Server 14

Scaling up an existing deployment 14

Managing Azure Database for PostgreSQL – Flexible Server using the Azure portal 14

Connecting to Azure Database for PostgreSQL – Flexible Server with Azure CLI and creating a database 14

HA features 15

Zonal (same zone) HA 15

Zone-redundant high availability 15

Security features 16

Data encryption 16

Regular updates/patching 16

User management and access controls 16

Threat detection, prevention, and mitigation 16

Backup and disaster recovery features 17

Automated and on-demand backups 17

Monitoring performance and health 17

Azure Monitor and alerts 17

Query performance insights 17

Building intelligent apps with Azure AI and Azure Database for PostgreSQL – Flexible Server 18

Azure integrations and database extensions 18

Troubleshooting and support 18

Azure support plans 18

Community resources 19

Documentation and knowledge base 19

IaaS deployments: PostgreSQL on Azure IaaS 20

Planning for your deployment on Azure IaaS 20

Azure VM series overview and operating system choices 21

Azure Virtual Machines series 21

AMD processor-based VM series to consider 21

OS compatibility 21

Azure Disk Storage overview 22

Azure Disk Storage types 22

Azure Disk Storage features 22

Azure Elastic SAN 23

Deploying PostgreSQL on Azure IaaS 25

Creating an Ubuntu VM 25

Installing PostgreSQL 16 on the Ubuntu VM 26

Compute considerations 27

VM sizing 27

CPU 27

Memory 27

Storage considerations 27

Configuring multiple volumes 28

OS considerations 29

OS options 29

Database considerations 29

PostgreSQL parameters 29

Connection configuration parameters 29

Memory configuration parameters 29

Write-ahead log configuration parameters 30

Query performance tuning 30

Scale considerations 31

Scaling up 31

Scaling out 31

HA considerations 31

Using managed disks 31

Availability zones 31

Availability sets 32

PostgreSQL 32

Replication 33

Streaming replication 33

Logical replication 33

Security considerations 33

Azure confidential computing 33

Identity Management 34

Azure Key Vault 34

Azure Storage 34

Disk security 34

Azure Networking 34

Azure Virtual Network 34
Azure DDoS Protection 35
Azure Firewall 35
Azure Load Balancer 35
PostgreSQL security features and considerations 35
Role-based access control (RBAC) 35
Row-level security 35
Authentication methods 36
Across the network 36
Encryption for data at rest 36
Encryption considerations 36
TCP/IP configuration 37
Firewall 37
Network segmentation 37
Backup considerations 37
Azure VM backup 37
File system backup 37
PostgreSQL backup considerations 38
Using pg_dump 38
Continuous archiving 38
Encrypting backups 38
Encrypting with OS tools 38
Encrypting backups with pgBackRest 38
Monitoring and maintenance 39
Azure Monitor 39
Monitoring at the PostgreSQL level 40
Other tools to consider 41
Percona PostgreSQL 41
Charmed PostgreSQL 41
Conclusion 42
Appendix 47
Sample ARM template for deploying Azure Database for PostgreSQL 47
template.json 47
parameters.json 52

# Introduction

PostgreSQL is one of the most used open-source relational databases in the market, and organizations can easily pair it with Microsoft Azure technologies to gain the flexibility and reliability benefits of the cloud. With its beginnings in the 1980s,[1] PostgreSQL has many features for handling structured and unstructured data, complex queries, and transactional or reporting applications. By choosing to run PostgreSQL on Azure, customers can enjoy the benefits of Azure, such as a myriad of VM choices, storage configurations, scalability, monitoring tools, security, and high availability options.

This document addresses deployment best practices for Azure Database for PostgreSQL – Flexible Server and PostgreSQL on Azure infrastructure as a service (IaaS). It covers VM selection, storage considerations, simple deployment procedures, monitoring and maintenance, database configuration considerations, and backup and restore.

## PaaS and IaaS: What to know for PostgreSQL deployments

As organizations continue to shift their PostgreSQL databases to the cloud, selecting the right deployment model becomes a critical decision that can impact flexibility, scalability, and operational efficiency. Two predominant cloud service models—Platform as a Service (PaaS) and Infrastructure as a Service (IaaS)—offer distinct advantages depending on the needs of an organization. PaaS streamlines application deployment by providing a ready-to-use platform that abstracts much of the underlying infrastructure complexity. IaaS provides a highly customizable cloud environment where businesses can have complete control over the operating system, storage, and networking.

By offering PaaS and optional infrastructure services for PostgreSQL, Azure allows organizations to select the model that best suits their needs while continuing to use familiar Microsoft and Azure tools. Choosing between the Azure PaaS and IaaS offerings comes down to the balance among organizations' needs for control, convenience, and available resources.

PaaS refers to cloud computing services that provide a managed platform for customers to develop, host, and manage applications without manually building, maintaining, and securing the underlying infrastructure. PaaS platforms deploy and configure infrastructure and automatically handle all maintenance, patching, and updates. With Azure Database for PostgreSQL – Flexible Server, this means that customers can focus on business-critical activities, including application development, instead of devoting resources to database management tasks, such as HA, disaster recovery, backups, and geographic data replication.

PaaS offers these specific advantages:

- **Less coding time**: Pre-coded application components such as workflows, directory services, security features, and search can boost app creation.
- **Faster time to market**: Access to cloud capabilities including development and devops tools to streamline the app dev process.
- **Easy scaling**: Scale up compute and storage to meet workload demands with the click of a button and near zero downtime while only paying for the resources you use.
- **Automated management and maintenance**: Azure handles HA, backups, patching, and provides intelligent tuning recommendations enabling organizations to reduce costs associated with administrative tasks while ensuring peak performance.

PaaS provides features that could bring additional advantages for staff and operations, including pre-coded application components such as workflows, directory services, security features, and search.

In contrast to the benefits of PaaS, IaaS demands more technical resources and effort for deployment and configuration but provides complete control over the OS and database. This level of control is crucial in scenarios such as hosting legacy applications or scenarios where there's a need for higher levels of customization.

IaaS offers the following specific benefits:

- Full control over the OS and database configuration: For compliance and regulatory contexts, IaaS solutions might be valuable. In these use cases and others, admins will likely need more granular control over the database or data and security.
- Flexibility: Customers running PostgreSQL on Azure IaaS can select which minor version of PostgreSQL they need. They also get more flexibility on when to apply patches and can configure all PostgreSQL parameters. In contrast, PaaS allow customers to edit many, but not all, PostgreSQL parameters.[2] Additionally, legacy applications might depend on configurations or software versions that modern PaaS solutions do not support.

IaaS provides flexibility and a greater degree of control over your infrastructure. However, manual deployment, management, and maintenance require higher operational overhead and technical knowledge. For organizations that require this level of control—or those that have in-house expertise—IaaS may be the right solution.

To make the right choice between PaaS and IaaS for your organization, you need to consider your specific needs and workloads. In this report, we explain how to plan and deploy either an Azure Database for PostgreSQL – Flexible Server solution or PostgreSQL on Azure IaaS.

# PaaS Deployments: Azure Database for PostgreSQL - Flexible Server

## Best practices for deploying Azure Database for PostgreSQL – Flexible Server

The section covers deployment procedures and considerations for Azure Database for PostgreSQL – Flexible Server, including compute selection, storage options, deployment methods, monitoring and maintenance considerations, failover and HA configurations, and backup and restore options.

### Understanding Azure Database for PostgreSQL – Flexible Server

Azure Database for PostgreSQL – Flexible Server is a fully managed, fully functional PostgreSQL database PaaS platform. Azure automates database deployment, management, maintenance, hosting, and HA tasks for the virtual infrastructure hosting the database. After deployment, users and applications can access the database service via a client of their choosing. Admins can adjust database parameters to match workload requirements and can scale compute and storage resources as needed. Azure also brings economy of scale to the database backend with robust networking, security, provisioning, and redundancy capabilities that might be costly and complex to maintain in a self-hosted environment.

Azure Database for PostgreSQL – Flexible Server supports PostgreSQL versions 11, 12, 13, 14, 15, and 16. Azure Database for PostgreSQL – Flexible Server does not support PostgreSQL versions 10 and earlier.[3] Azure automates regular patches and updates within the supported PostgreSQL versions on either a customer-defined or system-managed maintenance schedule of one maintenance window per week.[4]

**Benefits of deploying Azure Database for PostgreSQL – Flexible Server**

Deploying Azure Database for PostgreSQL – Flexible Server gives you all the benefits that PaaS solutions in the Azure ecosystem offer over traditional self-hosted database infrastructure and more. Azure benefits include automated maintenance, patching, and updates; enterprise-grade security from critical Azure and OpenSource tools, and the option to pay-as-you go. Additionally, you can quickly scale your compute and storage independently to meet increased workload demands while ensuring up to 99.99 percent SLA uptime. Multi-cloud customers can use Azure Arc-enabled PostgreSQL to gain benefits such as elastic scale, unified management, and a cloud billing mode for deployments outside of Azure.

With the latest AI capabilities supported by extensions like pgvector and integrations with other Azure services such as Azure AI Services, Azure Kubernetes Service and Azure App Service, Azure Database for PostgreSQL is an excellent choice for building generative AI applications. Extensions and tools for AI app development include a built-in extension for Azure AI that natively integrates with Azure AI Language, Azure Machine Learning, and other Azure AI services and Azure Cognitive Services to enable developers to build PostgreSQL-based generative AI apps and other AI-ready tools such as pgvector vector search.[5] To learn more about Azure Database for PostgreSQL – Flexible Server native AI-ready integrations and the Azure AI development ecosystem, see the Microsoft Learn training path on building AI apps with Azure Database for PostgreSQL – Flexible Server at https://learn.microsoft.com/training/paths/build-ai-apps-azure-database-postgresql/.

Scandinavian Airlines (SAS) provides a great example of how Azure Database for PostgreSQL – Flexible Server can help organizations. SAS switched to the Azure managed database service to lower infrastructure costs and gain agility for its development teams. Now the Airline's mission-critical booking app runs in Azure Kubernetes Service (AKS) backed by Azure Database for PostgreSQL – Flexible Server, freeing the app team from managing infrastructure so it can focus on innovation. Azure Database for PostgreSQL – Flexible Server provides zone-redundant high availability and burstable compute tiers to handle any spikes in traffic to the booking app, while also optimizing costs.[6]

Another example is Allego, a European electric vehicle charging company. The increasing demand for electric vehicles and the supporting charging infrastructure resulted in Allego's dramatic growth. When it came to making sure their core services were able to meet the data needs of the future, they chose to take advantage of the latest Azure database as a service offering by migrating to Azure Database for PostgreSQL – Flexible Server. The increases in scalability and manageability help to ensure that Allego customers will continue to experience reliable charging services, even as demand keeps growing.[7]

Principled Technologies previously conducted hands-on testing to demonstrate the ease and benefits of migrating an existing PostgreSQL database to Azure Database for PostgreSQL – Flexible Server on an AMD EPYC™ processor-based VM. Read the report at https://facts.pt/Y72pat3.

(Note: Azure Database for PostgreSQL – Flexible Server is "a fully managed database service designed to provide more granular control and flexibility over database management functions and configuration settings."[8] The offering has supplanted Azure Database for PostgreSQL – Single Server, which will retire by March 28, 2025.

## Planning for deployment

### Azure compute overview

You can create an Azure Database for PostgreSQL – Flexible Server instance in one of three service tiers, differentiated by the underlying Azure compute:

- B-series Burstable VMs
- D-series General Purpose VMs
- E-series Memory Optimized VMs

Within these service tiers, there are multiple VM options.

Azure VMs come in predefined groupings, or series of configurations. Within each series type, there are generations (e.g., v4, v5), and each VM size within the series has a certain number of virtual CPUs (vCPUs) and amount of virtual RAM. Azure documentation recommends certain families of VMs for specific application use cases. For example, Microsoft recommends E-series and D-series VMs for relational database applications.[9] To explore the different VM series types and their common use cases, see https://learn.microsoft.com/azure/virtual-machines/sizes.

When deploying PostgreSQL on Azure, you will determine the appropriate VM series and type for your specific use case. For production workloads, you will most likely select VMs from either the general purpose or memory optimized categories because PostgreSQL will heavily use CPU, RAM, or both. You are unlikely to need high performance computing.

### Azure VM service tiers and workload considerations

Table 1 provides detailed information on the three service tiers.

Table 1: Azure Database for PostgreSQL – Flexible Server VM sizing options. Source: Microsoft Azure.[10]

| | VM type | Number of vCores | Processor OEM | Target workloads |
|---|---|---|---|---|
| **Burstable** | | | | |
| | Standard_B*ms | 1, 2, 4, 8, 12, 16, 20 | Intel® | Web servers, development build environments, small databases |
| | Standard_B*s | 2 | Intel | |
| **General Purpose** | | | | |
| | Standard_D*ads_v5 | 2, 4, 8, 16, 32, 48, 64, 96 | AMD | Enterprise applications, such as SAP or Microsoft Dynamics, high traffic web servers, production-sized development environments |
| | Standard_D*ds_v5 | 2, 4, 8, 16, 32, 48, 64, 96 | Intel | |
| | Standard_D*ds_v4 | 2, 4, 8, 16, 32, 48, 64 | Intel | |
| | Standard_D*ds_v3 | 2, 4, 8, 16, 32, 48, 64 | Intel | |
| **Memory Optimized** | | | | |
| | Standard_E*ads_v5 | 2, 4, 8, 16, 20, 32, 48, 64, 96 | AMD | Data analytics, large in-memory relational databases, scientific simulations, data warehousing |
| | Standard_E*ds_v5 | 2, 4, 8, 16, 20, 32, 48, 64, 96 | Intel | |
| | Standard_E*ds_v4 | 2, 4, 8, 16, 20, 32, 48, 64 | Intel | |
| | Standard_E*s_v3 | 2, 4, 8, 16, 32, 48, 64 | Intel | |

B-series burstable VMs can be valuable for test or development workloads or use cases where baseline CPU utilization is low overall but may require periods of higher performance based on intermittent demand. In addition, burstable VMs do not offer HA.[11] Customers might also use B-series VMs for low-traffic web servers or applications, microservices, or other less critical workloads.

D-series general purpose compute VMs are the standard tier of Azure VMs. Customers might choose them for general production use cases and workloads. Microsoft recommends these VMs for enterprise customers with general purpose workloads that do not require a compute-to-memory ratio higher than 1:4 (for each vCore there is 4GiB memory). Examples of these workloads include e-commerce systems, production websites, hosted services, virtual desktop infrastructure hosting, and small to medium-sized databases.[12]

E-series memory optimized VMs "are optimized for in-memory applications"[13] and feature a higher than typical ratio of memory to compute power (up to 8GB of memory per vCore, compared to 4GB per vCore in the General Purpose VM service tier).[14] Microsoft recommends these VMs for workloads that require in-memory processing such as analytics, high-performance transactional workloads, exceptionally large databases, or business intelligence applications.[15]

### AMD processor-based VM series to consider

The two AMD processor-based VM series available for Azure Database for PostgreSQL – Flexible Servers are the Dadsv5 General Purpose tier VMs and the Eadsv5 Memory Optimized tier VMs. The Dadsv5 series provides a balanced ratio of compute to memory resources with 4 GiB of memory per virtual core, while the Eadsv5 series VMs feature 8 GiB of memory per vCore in most configurations.[16]

The Dadsv5 series features scalability up to 96 vCPUs and 384 GiBs of RAM, and AMD EPYC™ 7763v processors "that can achieve a boosted maximum frequency of 3.5GHz."[17] For more information on the Dadsv5 VMs, see https://learn.microsoft.com/azure/virtual-machines/dasv5-dadsv5-series.

Eadsv5-series VMs feature memory-heavy 8:1 RAM-to-vCPU ratios in most configurations,[18] with the smallest Eadsv5 VM having 16 GiBs of RAM with two vCores, and the largest VM configuration having 672 GiBs of RAM with 112 vCores. For more information on the Eadsv5 series, see https://learn.microsoft.com/azure/virtual-machines/easv5-eadsv5-series.

### Configurable options and limitations

Azure Database for PostgreSQL – Flexible Server instances have several provisioning and configuration options to consider prior to deployment, including service region, availability zone, authentication method, access scope, and HA. In addition to choosing an Azure region for the server, administrators can specify a particular availability zone within that region to collocate the database service with other Azure services or hosted applications that will use it. Customers can provision HA in either zone-redundant or same-zone configurations during the initial server deployment or enabled after the fact in the Azure portal. See the section High availability best practices for details.

Azure Database for PostgreSQL – Flexible Server instances support both traditional PostgreSQL authentication via a user-defined administrator username and password and Microsoft Entra authentication for accessing and administering the server. Customers can enable either or both depending on application and security requirements. Administrators can also control network access to the server in many granular ways. They can allow inbound connections from public IP addresses or whitelist specific IP addresses or ranges in the Azure firewall. They can also permit other Azure services to access the resource, or they can restrict access sources to specific private endpoints or Azure virtual networks. All Azure Database for PostgreSQL – Flexible Server deployments support encrypted TLS/SSL connections.

Due to its nature as a PaaS offering, Azure Database for PostgreSQL – Flexible Server does not offer the same level of operating system (OS) control as a self-managed IaaS deployment of PostgreSQL. If the configurable options detailed in this deployment guide are not sufficient for your workload or use case infrastructure requirements, you might find that an IaaS deployment works better.

### Storage considerations

Azure Database for PostgreSQL – Flexible Server sets maximum IOPS for each instance type, and the maximums differ for each instance type:[19]

- Burstable: 4,320 IOPS
- General purpose: 80,000 IOPS
- Memory Optimized: 80,000 IOPS

Administrators can set a pre-provisioned IOPS level during deployment or opt to let Azure auto-scale IOPS as needed. Administrators specify the initial storage size in GiB during deployment and can enable Azure to scale storage dynamically as needed to prevent interruption of service due to storage capacity. Administrators can also manually scale up the storage capacity at any time but cannot scale storage down after initially provisioning the server.

### Backup considerations

Administrators choose backup retention periods of 7 to 35 days for daily automated backups. By default, backups are zone redundant and customers can additionally choose to enable geo-redundancy during the deployment process. In addition to daily automated server backup snapshots, Azure backs up transaction logs at a workload-dependent rate with a recovery point objective of up to five minutes.[20]

## Deploying Azure Database for PostgreSQL – Flexible Server

This section details the process and configuration options for deploying Azure Database for PostgreSQL – Flexible Server via the Azure portal or the Azure CLI and discusses basic methods to connect to the PostgreSQL resource once deployed. Our examples detail a specific configuration we chose for our sample deployment, but we discuss all relevant configuration options in the step-by-step instructions below.

### Creating an Azure Database for PostgreSQL – Flexible Server via the Azure portal

1. **Navigate** to the Azure portal at https://portal.azure.com/ and **log in** using your Azure credentials.
2. Using the search bar, **navigate** to the **Azure Database for PostgreSQL – Flexible Servers** service **or select** it from the Azure services menu (see Figure 1).

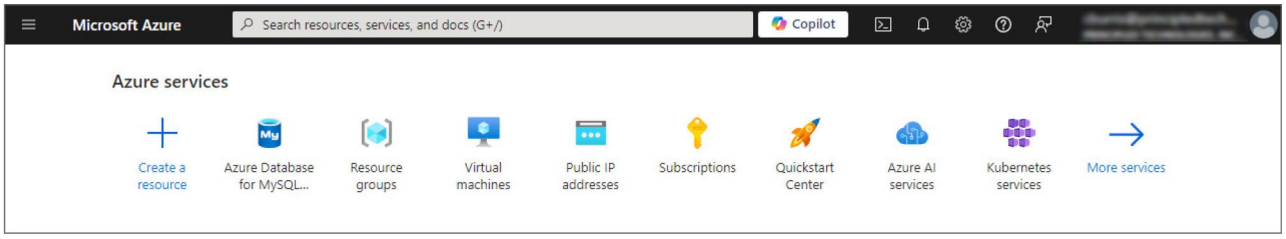



Figure 1: Screenshot of the Azure services menu after logging into the Azure portal. Source: Principled Technologies.

3. Click **Create**. Figure 2 shows the two locations of the Create button in the upper left and bottom center.

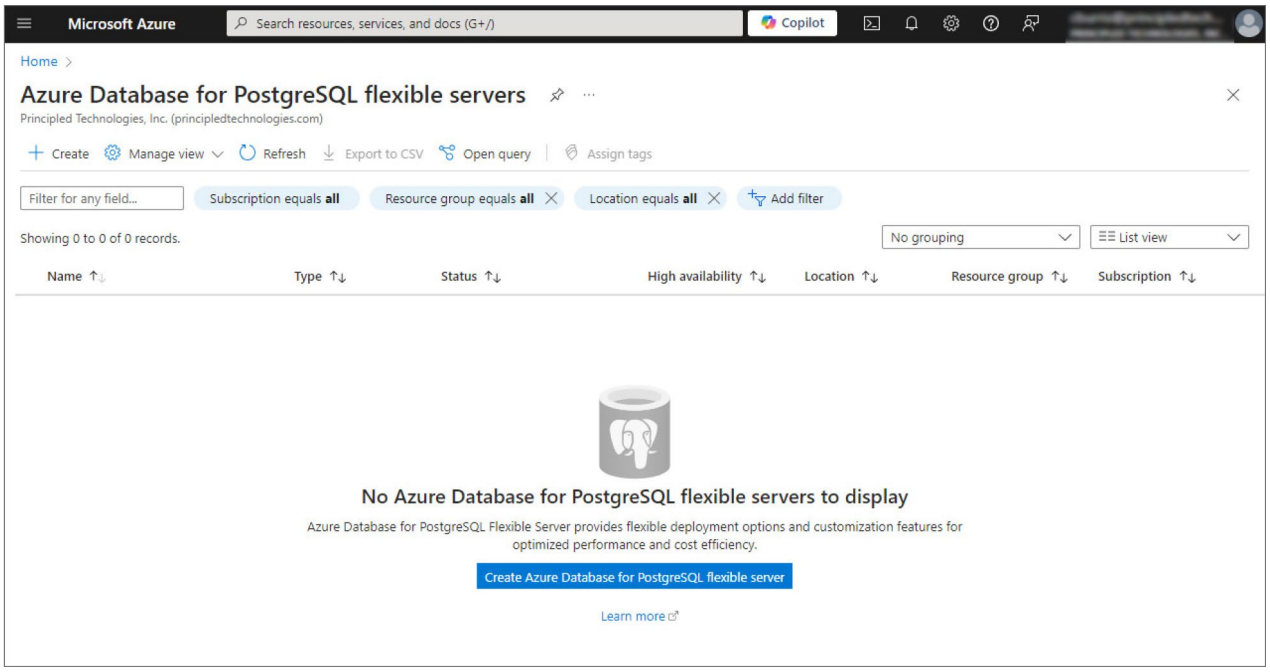



Figure 2: Screenshot of the Azure Database for PostgreSQL - Flexible Server management page in the Azure portal. Source: Principled Technologies

4. Under **Project Details**, **select** the Azure subscription and resource group to which you want to deploy the resource. To create a resource group, click **Create new** under the dropdown menu (see Figure 3).

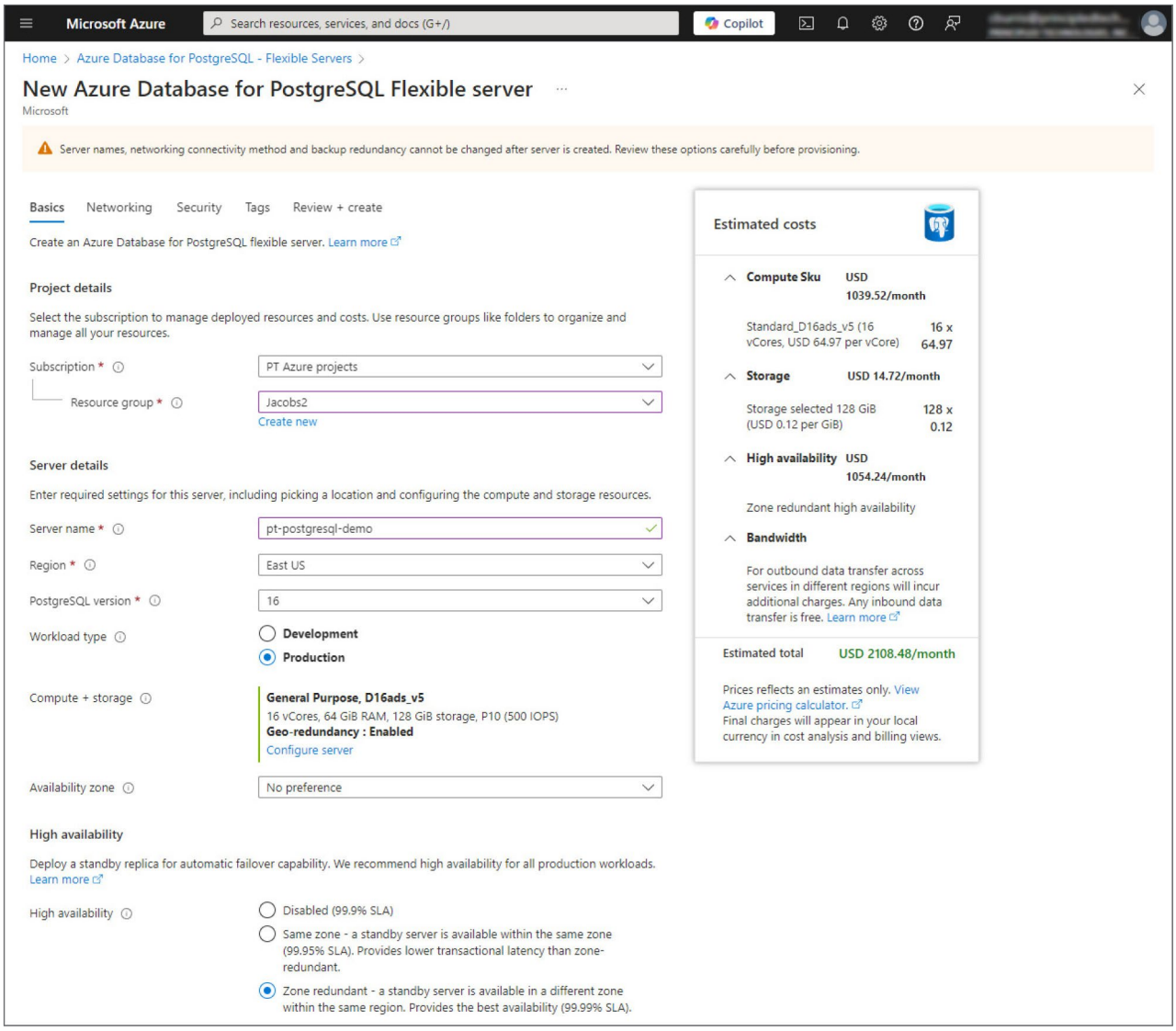



Figure 3: Screenshot of the Azure Database for PostgreSQL – Flexible Server creation page in the Azure portal. Source: Principled Technologies.

5. Under **Server details**:

   a. **Enter or select** the following details for the server:

   - **Server name** – a unique server name from 3 to 63 characters that may only contain lowercase letters, numbers, and hyphens; it cannot start or end with a hyphen
   - **Region** – any of the Azure regions (for our sample deployment, we chose East US)
   - **PostgreSQL version** – For our sample deployment, we chose PostgreSQL version 16.

b. **Specify** the server compute and storage details:

- **Select a Workload type.** This will determine which VM series and compute/storage resources the creation wizard recommends. "Development" defaults to B-series VMs, and "Production" defaults to D-series. For our sample deployment, we chose "Production."
- **Select Compute and storage** options by clicking Configure server. These options can give you more granular control over your compute, including tier, processor, and size, and allow you to adjust your needs for storage, HA, and backup. Figure 4 shows the Configure server page with our sample deployment values.

c. **Specify** the following options according to your workload needs:

i. **Compute tier** – Choose among Burstable, General Purpose, or Memory Optimized.

- For our sample deployment, we chose General Purpose

ii. **Compute processor** – Choose the instance that is appropriate for your workload.

- For our sample deployment, we chose an AMD EPYC™ instance. Note that only the General Purpose and Memory Optimized compute tiers offer AMD processors.

iii. **Compute size** – Choose one of the available VM series based on vCore, memory, and IOPS requirements.

- For our sample deployment, we chose the AMD CPU-based Standard_D16ads_v5 VM series with 16 vCores, 64 GiB memory, and 25,600 max IOPS.

iv. **Storage size** – Specify the size of available storage capacity to make available to the server in gibibytes (GiB). Customer can choose this value from a list of options ranging from 32 gibibytes (GiB) to 32 tebibytes (TiB), and the solution uses the capacity for database files and logs. Customers can add storage after creating the server, but they cannot scale down this value creation.

- For our sample deployment, we chose 64 GiB.

v. **Performance Tier** – Select the desired storage IOPS from the list.

- For our sample deployment, we chose P30 (5,000 IOPS).

vi. **Storage auto-growth** — Choose to enable storage auto-growth, which could increase your billable storage capacity if the server runs out of storage capacity and prevents the server from becoming read-only and interrupting workloads.

- For our sample deployment, we enabled storage auto-growth.

vii. **High availability** — Choose to enable HA. If customers enable HA, they can choose between same zone (zonal) HA or zone-redundant HA.

- For our sample deployment, we enabled zone-redundant HA.

viii. **Backup options** – Select a backup retention period in days (7 to 35) and enable geo-redundancy. Note that users cannot enable geo-backups after server creation. Users cannot enable geo-redundancy after server creation when using zone-redundant HA mode.

- For our sample deployment, we specified a 14-day backup retention period and enabled geo-redundancy.

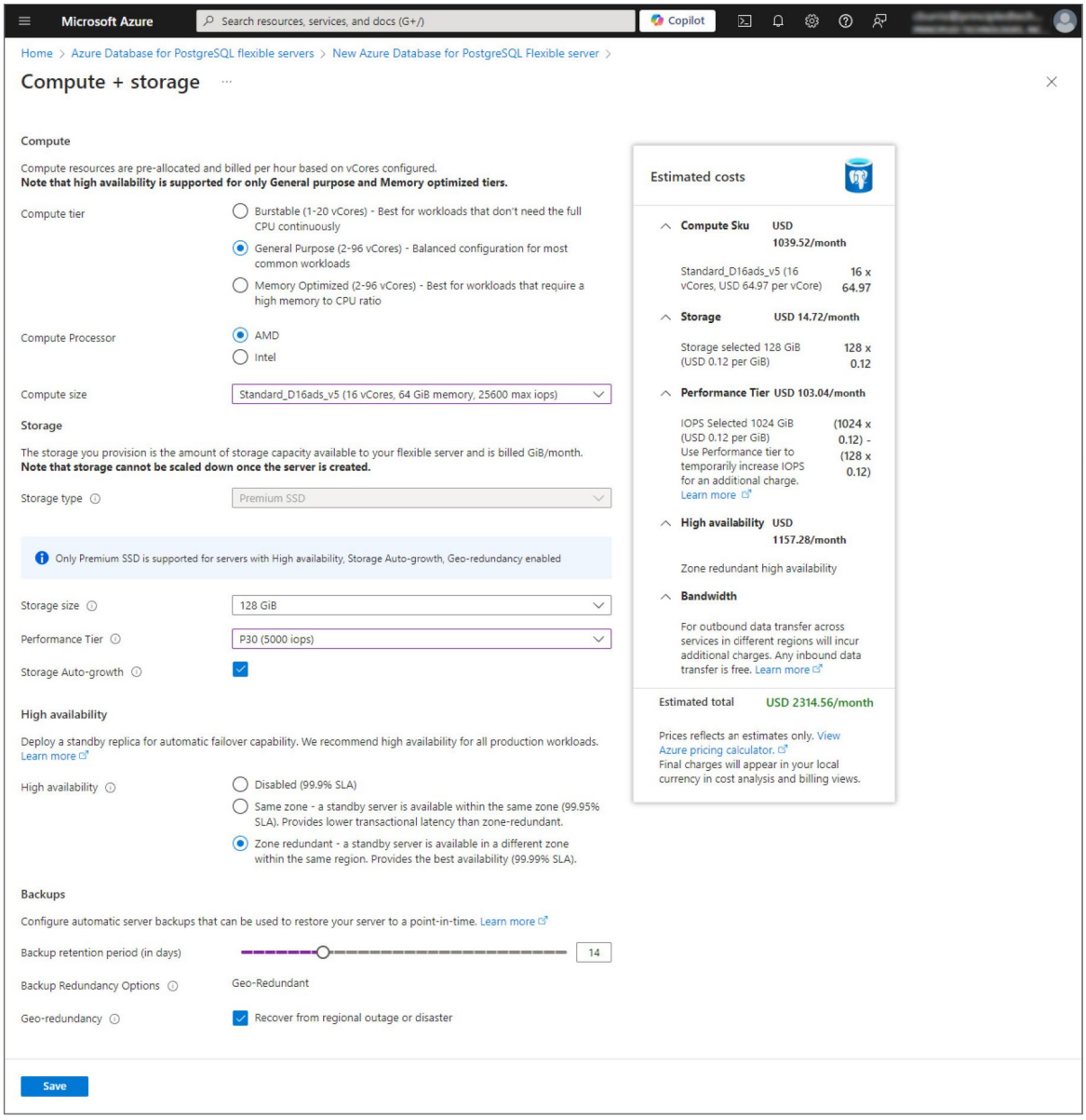



Figure 4: A screenshot of the Configure server page showing values for our sample deployment. Source: Principled Technologies.

d. After configuring your compute and storage options, **click Save**.

e. If desired, **select** a specific **Availability zone** for deployment. This might be useful if you want your database to exist in a specific availability zone that is co-located with an application or workload that uses it.

   i. For our sample deployment, we chose no preference.

f. Under **High availability, choose** your desired setting.

g. Under **Authentication**

- **Choose Authentication method**: PostgreSQL authentication, Microsoft Entra authentication, or both.
  - For our sample deployment, we chose PostgreSQL authentication.
- **Enter an Admin username and password** for the server. If you are using Microsoft Entra authentication, select a user-assigned managed identity and a Microsoft Entra admin.

6. **Click Next → Networking.**

   a. **Select connectivity method:** public or private access. Public access enables you to allow access to the server from all public IP addresses, to only other Azure services, to only a specific range of public IPs, through a private endpoint via an Azure virtual network, or any combination of the four. Private access limits access through using a private endpoint via Azure virtual network integration.

      i. For our sample deployment, we chose public access.

   b. If using **public access**:

      i. Choose to allow **public access** to this resource through the internet from a public IP address. For our sample deployment, we did not enable this option.

      ii. Choose to allow public access from any Azure service within Azure to this server. For our sample deployment, we enabled this option.

      iii. Enter a **firewall rule** name and start and end IP addresses to give a range of public IP addresses access to the server if applicable. For our sample deployment, we added a rule to allow access to only our office public IP address.

      iv. To allow private access, create a private endpoint by clicking **Add private endpoint** and specifying a virtual network, subnet, and private DNS zone.

   c. If using **private access** (VNet integration):

      i. Select the Azure **subscription, virtual network, and subnet** to use for private access. Azure will create a virtual network and subnet in the same resource group by default if unspecified.

      ii. Select a **private DNS** zone or leave blank to automatically create one.

7. **Click Next → Security.**

   a. Azure encrypts the storage by default using service-managed keys. If you would like to use a customer-managed key instead, select the user-assigned managed identity and key here.

8. **Click Next → Tags.**

   a. Enter any Azure tags based on organizational requirements that you wish to associate with the Azure Database for PostgreSQL – Flexible Server.

9. **Click Next → Review** and create. This screen validates the server configuration and estimates the server creation time in minutes. Click Create to confirm and initiate server deployment. Figure 5 shows a successful Azure Database for PostgreSQL – Flexible Server deployment using the Azure portal.

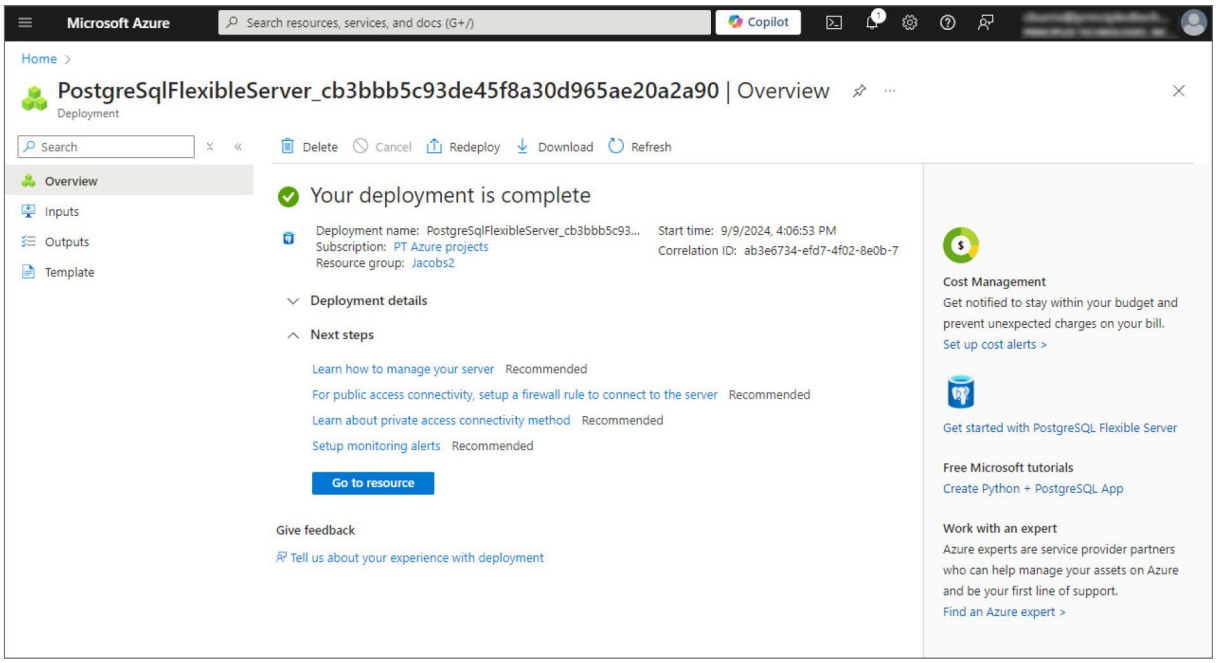



Figure 5: A screenshot of a successful Azure Database for PostgreSQL deployment. Source: Principled Technologies.

## Creating an Azure Database for PostgreSQL – Flexible Server via Azure CLI

Administrators can also deploy Azure Database for PostgreSQL – Flexible Server directly from the command line using Azure Cloud Shell or Azure CLI. We used Azure Cloud Shell for our sample deployment, but the following steps should work with an authenticated user on Azure CLI version 2.0 or later as well.

1. **Open** Azure Cloud Shell by logging into the Azure portal and navigating to https://portal.azure.com/#cloudshell/. Select Bash if prompted.
2. **Select** your active Azure subscription if prompted and click Apply.
3. If desired, **create** a new resource group for deployment:

```
az group create --name sampleresourcegroup --location eastus
```

4. **Create** a new Azure Database for PostgreSQL server by running the following command and modifying the parameters and values as needed. Any parameters without values will revert to Azure CLI defaults.

```
az postgres flexible-server create --name samplepostgresqlserver
--resource-group sampleresourcegroup
```

5. **Respond** yes or no to any prompts that appear during the deployment process. Azure will restrict the server to private access by default, which will prompt you to allow your client IP address to connect. We enabled this in our sample deployment.
6. After the cloud shell console indicates that you have successfully deployed and configured the server, it

will display a short JSON object output that shows the server connection string, associated identifiers and resources, the VM series SKU, and the generated admin username and password if you did not specify them in step 4. The server is now ready to use.

7. To output the server details and connection information in JSON format and save the output for reference, **run** the following command:

```
az postgres flexible-server show --resource-group sampleresourcegroup
--name samplepostgresqlserver
```

8. **Run** the following command to test your connection to the database and confirm your client connects to the database server as expected, replacing <username> and <password> with the values from the JSON output in the previous step:

```
az postgres flexible-server connect -n samplepostgresqlserver -u
<username> -p <password>
```

### Overview of using Azure Resource Manager templates

Azure Database for PostgreSQL – Flexible Server supports provisioning via Azure Resource Manager (ARM) templates. An ARM template is a JSON file that defines the infrastructure and configuration for a deployment, which could consist of multiple databases on a single server or across multiple servers. ARM templates enable customers to automate provisioning and deployment of Azure Database for PostgreSQL – Flexible Server instances with the same configuration options that users can specify when manually deploying instances via the Azure portal.

See the appendix for an example ARM template that you can use to deploy an Azure Database for PostgreSQL – Flexible Server instance based on our sample configuration.

## Managing Azure Database for PostgreSQL – Flexible Server

### Scaling up an existing deployment

Users can scale up or down Azure Database for PostgreSQL – Flexible Server compute resources by adjusting the compute size in the Azure portal. Users can also scale up storage, but not scale it down. Changing the VM type will require a server restart. You can follow these steps to scale up an existing Azure Database for PostgreSQL – Flexible Server in the Azure portal:

1. **Log into** the Azure portal and navigate to the Azure Database for PostgreSQL – Flexible Server service using the services list or the search bar.
2. **Click** the name of the Azure Database for PostgreSQL – Flexible Server instance to view its Overview page.
3. **Click** Settings in the left navigation pane to expand the settings and click Compute + storage.
4. **Adjust** the compute tier, compute size (VM series) and storage size, IOPS, and auto-growth setting as needed. You can also enable post-deployment HA here or modify the backup retention period. The estimated costs panel at the bottom of the page reflects the new estimated monthly server cost based on configuration changes.
5. **Click** Save, and then when prompted to restart the server, click Continue.

### Managing Azure Database for PostgreSQL – Flexible Server using the Azure portal

Users can manage Azure Database for PostgreSQL – Flexible Server primarily using the Azure portal. To manage your server, select your Azure Database for PostgreSQL – Flexible Server instance in the Azure portal and open the Overview panel by clicking the Overview menu. Then if you click Databases, you can click Add to create a new database on the server, which allows you to specify the database name, collation, and character set. You can also click Delete to remove a database from the server.

Administrators can also modify PostgreSQL server parameters by clicking Server parameters in the left-side menu. There you can view all server parameters and modify them where possible. Modifying static parameters will prompt you to restart the server either immediately or later. This management option also allows you to reset all server parameters to default if they are misconfigured. For more detailed information about modifying PostgreSQL server parameters, see https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/how-to-configure-server-parameters-using-portal.

The settings section of the menu also allows administrators to configure or enable replication, HA, backup settings, security settings, alerts and monitors, and other basic configuration options after server deployment.

#### Connecting to Azure Database for PostgreSQL – Flexible Server with Azure CLI and creating a database

1. **Open** the Azure Cloud Shell or Azure CLI and connect to the server:

```
az postgres flexible-server connect -n samplepostgresqlserver -u
<username> -p <password>
```

2. **Create** a new database and specify the database name, the server name, and the resource group. If successful, you will see confirmation output consisting of database details in JSON format.

```
az postgres flexible-server db create -d sampledbname --server-name
samplepostgresqlserver --resource-group sampleresourcegroup
```

3. **Confirm** deployment by connecting to the newly created database:

```
az postgres flexible-server connect -n samplepostgresqlserver -u <username>
-p <password> -d sampledbname
```

4. **Run** the following command to test with a database query if desired:

```
az postgres flexible-server connect -n <server-name> -u <username> -p
<password> -d <database-name> --querytext "<query text>"
```

For more information about the connect command, see the Azure CLI documentation at https://learn.microsoft.com/azure/postgresql/flexible-server/connect-azure-cli.

### HA features

Azure Database for PostgreSQL – Flexible Server provisions physically separated primary and standby replicas in the same availability zone (zonal) or across availability zones (zone-redundant). Both configurations allow automatic failover with zero data loss.

Availability zones are separate data centers within the same region. Each data center has independent power, cooling, and networking. Azure services move to the nearest working availability zone in the event of a failure.

Azure always deploys standby replicas in the same VM configuration, and customers can remove them by disabling HA. Azure performs stop, start, and restart operations on both primary and standby servers at the same time.

Azure applies maintenance activities, such as updates, to the standby server first, and then promotes the standby server to primary while Azure updates the original primary server. This provides continuous uptime.

### Zonal (same zone) HA

Zonal redundancy (primary and standby replicas in the same availability zone) provides redundancy with lower latency than the zone-redundant configuration. Zonal redundancy could protect from node-level failures while reducing downtime during planned and unplanned events.

Azure backs up data and logs onto local redundant storage. Admins can choose the region and availability zone for deployment. Zonal redundancy offers an uptime SLA of 99.95 percent.[21]

#### Zone-redundant high availability

Zone redundancy (primary and standby replicas in different zones) provides the highest level of availability at the cost of some latency. The latency will show at the application level but is necessary to keep replicas up to date.[22] According to Azure, the latency will not affect read-only queries, but customers can expect a 20 to 30 percent impact for writes and commits.[23]

Zone redundancy provides complete physical separation between the primary and standby stacks. Azure mirrors data and logs three times onto zone-redundant storage for additional data protection.

Admins can choose the region and availability zones for both primary and standby locations. Zone-redundancy offers an uptime SLA of 99.99 percent.[24]

Note: The Burstable compute tier does not support high availability.

## Security features

### Data encryption

Azure encrypts server, database, and backup data with a service-managed key by default, but customers can choose a user-managed key for encryption instead. Azure Database for PostgreSQL – Flexible Server integrates with Microsoft Entra and Azure Key Vault to create an Azure-wide authentication method. Role-based access and user-managed identities give admins flexibility in managing data access. Customers can use customer-managed or service-managed keys to encrypt their Azure Database for PostgreSQL – Flexible Server instances and backups. Azure stores the customer-managed key in an Azure Key Vault instance, which is highly available and scalable. Customers also have the option to have FIPS 140-validated hardware security modules back Azure Key Vaults. The Azure Key Vault and Azure Database for PostgreSQL – Flexible Server service must belong to the same Microsoft Entra tenant.[25]

### Regular updates/patching

Azure Database for PostgreSQL – Flexible Server undergoes regular maintenance to ensure the managed database remains secure, stable, and current. During these maintenance periods, the server receives new features, updates, and patches. The standard maintenance cycle occurs at least every 30 days. Admins have the option to schedule maintenance on a specific day and time window of their choice, or they can allow the system to automatically select a day and time window for you. If you do not specify a preference, the system will choose a time between 11 PM and 7 AM in the server's regional time zone.[26]

### User management and access controls

PostgreSQL and Microsoft Entra authentication is the default method of connecting to Azure Database for PostgreSQL — Flexible Server. However, admins can select either authentication as the sole method of connection.[27]

Admins can restrict network access via allow list (public IPs) or use Azure tools to create a private network for access control.[28]

### Threat detection, prevention, and mitigation

At an additional cost, admins can deploy Microsoft Defender for open-source relational databases to help monitor, investigate, and mitigate attacks. Microsoft Defender for open-source relational databases integrates with Microsoft Sentinel. After deploying Defender, the following activities will trigger alerts:

- Anomalous database access and query patterns
- Suspicious database activities
- Brute-force attacks

See the full list of alerts at https://learn.microsoft.com/azure/defender-for-cloud/alerts-open-source-relational-databases.

## Backup and disaster recovery features

### Automated and on-demand backups

Administrators choose backup retention periods of 7 to 35 days for daily automated backups and can specify whether backups are locally redundant, zone-redundant (for zone-redundant HA deployments), or geographically redundant in paired Azure regions. In addition to daily automated server backup snapshots, Azure takes transaction log backups every five minutes although this interval may vary based on workload requirements. If a daily backup fails for any reason, Azure retries the backup every 20 minutes until successful.[29]

## Monitoring performance and health

### Azure Monitor and alerts

Azure Monitor monitors Azure Database for PostgreSQL – Flexible Server. All Azure metrics have a one-minute frequency with 30 days of history. Monitor pulls metrics, such as CPU and storage performance data, from Azure and the database. Admins can place metrics on a dashboard with various charts and alerts.[30] See the full list of metrics at https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-monitoring.

Some Azure Database for PostgreSQL – Flexible Server metrics:

- Performance data
    - CPU, memory, and storage utilization percentages
    - Uptime
    - Credit usage (for burstable instance types)
- Storage breakdown
    - Capacity and limits
- Replication
    - Replication lag and status
- DML statistics
    - Select, update, insert, and delete statement counts

Admins can configure server database logging using the Azure portal and CLI by modifying the relevant database parameters. Admins can choose to send logs to Azure Monitor Logs, Azure Storage, or Azure Event Hubs automatically.[31]

Because Azure Database for PostgreSQL – Flexible Server integrates with Azure Monitor, admins can use Azure workbooks to parse and display metrics. Azure Database for PostgreSQL – Flexible Server comes with two workbook templates: overview and enhanced metrics. Admins can edit and customize templates as needed.

### Query performance insights

Admins can enable Query Store (which Azure disables by default) to help troubleshoot slow queries with Query Performance Insight. Admins can access Query Performance Insight via the Azure portal.

## Building intelligent apps with Azure AI and Azure Database for PostgreSQL – Flexible Server

### Azure integrations and database extensions

Azure Database for PostgreSQL – Flexible Server offers turnkey integration with a variety of Azure application development and AI services. The integrations enable developers to use the capabilities of the Azure development ecosystem and Azure AI to build and iterate their PostgreSQL-backed applications and workloads with less time to market while freeing resources previously devoted to infrastructure management. Azure Database for PostgreSQL – Flexible Server also offers built-in integration with Azure AI Services, Azure Kubernetes Service, Azure App Service, Power BI, and more. The Azure Database for PostgreSQL – Flexible Server extension for Azure AI enables seamless integration with Azure AI services such as Azure Cognitive Services and Azure OpenAI to use Generative AI and large language models (LLMs) in PostgreSQL-backed applications.[32]

Also core to the AI capabilities of Azure Database for PostgreSQL – Flexible Server is support for the pgvector database extension, an open-source extension that adds vector similarity search capabilities to PostgreSQL.[33] Vector similarity search is a fundamental component of many AI and machine learning applications that can provide efficient search, comparison, and analysis of embedded vector representations of various types of data.[34] This provides AI app and model developers with a powerful tool to enhance search and recommendation engines and other AI or AI-adjacent use cases.

## Troubleshooting and support

### Azure support plans

Azure offers four tiers of support plans. All customers have the basic Request Support tier, which includes access to Azure documentation and community resources, the ability to submit support tickets, access to the Azure Advisor service, and Azure health status notifications.

Azure offers the Developer support tier at $29 per month. Azure intends for this tier to support test and development environments. The tier includes all of the resources in the basic tier with the addition of "third-party software support with interoperability and configuration guidance and troubleshooting" and offers full access to Azure support by email during business hours after customers initiate a support request.[35] Azure limits developer support tickets to severity level C (minimal business impact) and have a guaranteed response time of 8 business hours.

Next is the Standard support tier at $100 per month. Azure intends for this tier to support production workloads and applications. It includes all resources and features of the basic and developer tiers with the addition of 24/7 access to technical support by phone or email. The tier also includes ticket creation with severity level B (moderate business impact) and level A (critical business impact) with response times of within 4 hours and 1 hour, respectively.[36]

Azure offers the Professional Direct support tier for $1,000 per month. Azure intends for this tier to support business-critical workloads and applications. This tier includes all the features of the other tiers but with shorter response times: 4 business hours for severity level C, 2 hours (regardless of business hours) for severity level B, and 1 hour for severity level A. The tier also provides access to a pool of ProDirect delivery managers for architectural and operational support and service review and advisory consultation and guidance.[37]

For more detailed information about the Azure support plans available, visit https://azure.microsoft.com/support/plans/.

### Community resources

In addition to Azure support, there are several community resources available to supplement the official documentation and support resources and connect users with other Azure administrators and Microsoft engineers:

- Azure community support – Customers can find the landing page for the following resources at https://azure.microsoft.com/support/community/.
- Azure on Q&A (Microsoft community support forum) – Users can create a thread to ask questions about specific Azure services and get feedback or support from other community members and Microsoft experts at https://learn.microsoft.com/answers/tags/133/azure.
- Azure on Stack Overflow – Stack Overflow, a well-known third-party community resource for technical support, has an officially supported community channel at https://stackoverflow.com/collectives/azure.
- Azure Support on Twitter – Users can also log onto the social media site X (formerly known as Twitter) to engage with Azure experts at https://x.com/AzureSupport.
- Azure Database for PostgreSQL Documentation – Explore the official documentation for Azure Database for PostgreSQL at http://aka.ms/postgresdocs.
- Azure Database for PostgreSQL Blog – Stay up to date with the latest news and features for Azure Database for PostgreSQL by visiting the official blog at https://aka.ms/azurepostgresblog.
- Azure Postgres Feedback Forum – Customers can share feedback and suggestions for Azure Database for PostgreSQL at https://aka.ms/pgfeedback.

### Documentation and knowledge base

Azure Database for PostgreSQL – Flexible Server has extensive official documentation, tutorials, and quick start guides for various common tasks and use cases:

- The official comprehensive documentation for Azure Database for PostgreSQL – Flexible Server is available at https://learn.microsoft.com/azure/postgresql/.
- Microsoft Learn includes a series of training tutorials for common use cases with Azure Database for PostgreSQL – Flexible Server that users can access at https://learn.microsoft.com/training/paths/microsoft-learn-azure-database-for-postgresql/.
- Azure offers a library of how-to technical training videos produced by Microsoft experts that cover various aspects of using Azure services and the Azure portal at https://azure.microsoft.com/resources/videos.

# IaaS deployments: PostgreSQL on Azure IaaS

## Planning for your deployment on Azure IaaS

Whether on site or in the cloud, a PostgreSQL deployment requires careful consideration of several factors, which we discuss in this report and list below. Each environment and application is different, so this list is not exhaustive; you should consider your environment's specifics in your planning.

**Storage:** Azure offers several varieties of storage, each with a range of capabilities, that administrators can attach to virtual machines. Workloads that create a small block random input/output (IO) pattern, such as most traditional online transaction processing (OLTP) workloads, may require virtual disks backed by high-speed SSDs that have higher input/output operations per second (IOPS) ratings. On the other hand, if your PostgreSQL data is mostly at rest or does not need significant bursts of random IO, it may be best to consider more cost-effective, lower-IOPS options.

**VM selection and configuration:** Azure also provides many VM series with a range of specifications and costs. Azure has optimized each series for different use cases and workloads, so you can balance cost and performance according to your needs. You should carefully configure the VM you choose from the perspective of virtual storage and operating system software. Your choice of VM type will dictate most options, such as core count and virtual RAM, but you might consider configuring multiple volumes if you require a high degree of customization. For operating system choices, nearly any Linux distribution will work with PostgreSQL. However, you'll likely want to use your organization's standard OS for compliance reasons, and you should check the PostgreSQL release version's compatibility with the Linux distribution version you choose at https://www.postgresql.org/download/linux/.

**Database and VM sizing:** To select the appropriate Azure VM and disks, you will need some knowledge of your databases' sizes. You should consider the current sizing of all elements of the PostgreSQL database application, such as tables, indexes, and log files. You should also consider their projected growth over time. We discuss this more in depth later in this document.

**PostgreSQL configuration:** PostgreSQL is an advanced database management system (DBMS), with many configuration options that you can modify to affect application performance, such as cache settings, IO concurrency, logging settings, and checkpoint behaviors.

**Networking:** Azure offers robust networking options. You'll want to be sure to explore the necessary networking options to set up the virtual networks you need, while controlling access to your PostgreSQL database.

**Monitoring and maintenance:** While the PaaS offering from Microsoft includes patching and updates, the IaaS PostgreSQL implementations are customer-supported and customer-patched. You should consider the maintenance needs of your PostgreSQL implementation up front. In addition, you must monitor your database application, and Azure offers myriad options to do so. Combining these Azure Monitor tools with traditional PostgreSQL monitoring approaches yields many options for a stronger implementation.

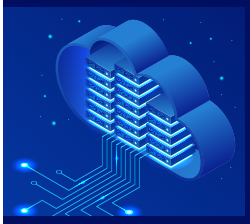
**Security:** Security is critical in any implementation, but especially on a cloud implementation. In this section, we will review various security best practices for both Azure and PostgreSQL.

**Backup and restore:** Any database implementation needs a carefully planned backup and restore framework. Later in this document, we discuss how to approach this critical piece of the puzzle in the context of running PostgreSQL on Azure in an IaaS implementation.

## Azure VM series overview and operating system choices

### Azure Virtual Machines series

Azure Virtual Machines come in predefined groupings, or series of configurations. Microsoft has designed each series for certain use cases. Within each series type, there are generations (e.g., v5, v6), and each VM type within the series has a certain number of virtual CPUs (vCPUs) and amount of virtual RAM. Azure documentation recommends certain families of VMs for certain application use cases. For example, they recommend E-series and D-series VMs for relational database applications. To explore the different VM series types and their common use cases, see https://learn.microsoft.com/azure/virtual-machines/sizes.



When deploying PostgreSQL on Azure, you will determine the appropriate VM series and type for your specific use case. Most likely, you'll select VMs from either the compute optimized or memory optimized categories because PostgreSQL will heavily use CPU, RAM, or both. You could consider VMs within the general purpose or storage optimized series, with general purpose being for development or light-use databases, and storage optimized for high-disk-throughput databases.

### AMD processor-based VM series to consider

There are three recommended AMD processor types: the Easv6 and Eadsv6 memory-optimized VMs and the Famsv6 compute-optimized VMs. Both feature sizeable RAM-to-vCPU ratios, making them excellent candidates for database workloads where you need much of your working dataset cached in database buffers.

The Easv6 series and Eadsv6 series both feature scalability up to 96 vCPUs and 672 GiBs of RAM, and AMD EPYC™ 9004 processors "that can achieve a boosted maximum frequency of 3.7GHz."[38] The Eadsv6 series VMs also include a local NVMe temp disk for application and system short-term storage, which is helpful for page files, swap files, and SQL tempdb files, among other things. For more information on the Easv6 VMs, see https://learn.microsoft.com/azure/virtual-machines/easv6-eadsv6-series. For more information on the Easdv6 VMs, see https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/memory-optimized/eadsv6-series.

Famsv6 VMs feature full cores (no simultaneous multithreading) and memory-heavy 8:1 RAM-to-vCPU ratios,[39] with the smallest Famsv6 VM having 16 GiBs of RAM, and the largest VM having 512 GiBs of RAM. For more information on the Famsv6 series, see https://learn.microsoft.com/azure/virtual-machines/fasv6-falsv6-series.

In addition to Famsv6 VMs, we also recommend Basv2, Falsv6 VMs, Fasv6, and Da-series v6 VMs for very small or non-production databases.

## OS compatibility

PostgreSQL is compatible with many commonly available operating systems and is likely to run on all the operating systems the Azure platform offers. PostgreSQL documentation states that "PostgreSQL can be expected to work on current versions of these operating systems: Linux, Windows, FreeBSD, OpenBSD, NetBSD, DragonFlyBSD, macOS, AIX, Solaris, and illumos."[40] For this guide, we chose Ubuntu as the underlying operating system.

## Azure Disk Storage overview

Because persistent and consistent data handling is a cornerstone of a reliable database application, storage design plays a key role in any database deployment. When designing in the context of Azure, you must consider Azure Disk Storage offerings and how to present those to the guest operating system, and thus, the database. In this section, we explore the Azure Disk Storage types and their features.

### Azure Disk Storage types

Azure Disk Storage is a service for managing Azure VMs' virtualized disk storage. Azure manages the underlying storage hardware, eliminating the need for users to maintain infrastructure. Users can simply choose a size and disk type, provision the disk, and then attach the disk to a VM.

Azure offers five disk storage types: Ultra Disk, Premium SSD v2, Premium SSD, Standard SSD, and Standard HDD. Table 1 shows the specifications for each as well as examples of workloads for which each disk type is suited. Note that while these limits apply specifically to Azure Disk Storage, Azure VMs also have their own unique storage performance limits depending on the VM series, as we discuss below.

Table 2: Comparison of storage performance maximums and workload scenarios for the disk types available via Azure Disk Storage. Source: https://learn.microsoft.com/azure/virtual-machines/disks-types.

| | Ultra Disk | Premium SSD v2 | Premium SSD | Standard SSD | Standard HDD |
|---|---|---|---|---|---|
| Disk type | SSD | SSD | SSD | SSD | HDD |
| Scenario | IO-intensive workloads such as SAP HANA, top tier databases (for example, SQL, Oracle), and other transaction-heavy workloads | Production and performance-sensitive workloads that consistently require low latency and high IOPS and throughput | Production and performance sensitive workloads | Web servers, lightly used enterprise applications and dev/test | Backup, non-critical, infrequent access |
| Max disk size | 65,536 GiB | 65,536 GiB | 32,767 GiB | 32,767 GiB | 32,767 GiB |
| Max throughput | 10,000 MB/s | 1,200 MB/s | 900 MB/s | 750 MB/s | 500 MB/s |
| Max IOPS | 400,000 | 80,000 | 20,000 | 6,000 | 2,000, 3,000* |
| Usable as OS Disk? | No | No | Yes | Yes | Yes |

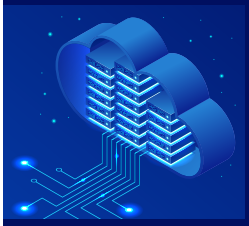
*Only applies to disks with performance plus (preview) enabled.

### Azure Disk Storage features

These disks come equipped with several benefits and features.

**Storage availability and durability:** Azure has designed managed disks for at least 5 nines (99.99999 percent) of availability via high levels of durability and redundancy.[41] Azure achieves this redundancy by creating three replicas of your data, meaning that even in the event of one or two replicas failing, the remaining replicas can protect against data loss. In addition, locally redundant storage (LRS) disks provide at least 11 nines (99.99999999999 percent) of durability each year, and zone redundant storage (ZRS) disks provide at least 12 nines (99.999999999999 percent) of durability each year. Azure Disk Storage is integrated with both availability sets and availability zone features provided by Azure, providing further protection for user data. Availability set integration ensures that disks associated with VMs in an availability set are isolated from each other by placing the disks in different storage scale units known as stamps, thereby avoiding a single point of failure. Similarly, availability zone integration protects against a single point of failure by giving users the ability to place disks in data centers in separate physical locations.[42]

**Storage scalability and using multiple volumes:** Users can create up to 50,000 VM disks of a particular type per region in a subscription. Disks can range in capacity from 4 GiBs to 64 TB, providing performance specs of anywhere from 120 to 400,000 IOPS and 25 MB/s to 10,000 MB/s of throughput, allowing you to customize the performance profile of your storage based on your needs.



In many cases, especially development or prototyping situations, it is possible to place data and log files on the same virtual disks. In more complex situations with larger databases or when optimizing for performance, we recommend splitting the data files and write-ahead logging (WAL) files onto separate volumes due to their differing IO types and to benefit from host caching. If you choose to use multiple volumes, you can configure these at instance creation or after the fact.

For the data drive, consider using Premium SSD v2 disks which provide additional features, high IOPS, and high throughput for a competitive price. For the log drive, plan for capacity projections. On all disk types, you should thoroughly test performance and evaluate costs. In general, the Premium SSD v2 disks or the Premium P30-P80 disks are good choices for high IOPS. If extremely low latency is required, also consider evaluating Azure Ultra Disks. Note that Azure offers Premium SSD v2 disks in only certain regions.

**Backup support for disks:** Managed disks include support for Azure Backup, a service that can create a backup job with scheduled backups and customizable retention policies. Additionally, Azure Backup includes Azure Disk Backup, a native solution that automates periodic creation of point-in-time snapshots for managed disks and retention of said backups based on user-defined policy. It supports backups for both OS and data disks, regardless of whether they are currently attached to a running Azure VM.[43] It is an agentless solution, meaning that the snapshot process does not affect application performance.

**Snapshots and images:** You can take a snapshot of your managed disks, creating a read-only full copy of the disk that exists independently of the source disk. You can use these to create new disks and to take a backup of a disk at any time. Azure takes snapshots from a single disk and stores them as standard managed disks by default. Azure incremental disk snapshots are a cost-effective way to create point-in-time backups of your Azure managed disks. Unlike full disk snapshots, which capture full disk data, incremental snapshots capture only the changes to disks since the last snapshot. This results in lower storage costs and faster snapshot creation times.[44]

Azure Disk Storage also supports the creation of custom images from either a user-managed VHD in a storage account or directly from a generalized VM. The capture process includes all managed disks attached to a VM, creating a single image. You can use these images to clone VMs that you have configured beyond the standard OS images provided by Azure.

### Azure Elastic SAN

Azure Elastic SAN is a service Azure provides that allows you to create and provision a cloud-based SAN interoperable with Azure VMs, as well as other Azure offerings such as Azure Kubernetes Service and Azure VMware Solutions.[45] Azure Elastic SAN is not limited to working with one particular type of storage-consuming host or service. Additionally, you can provision volumes from a single SAN to use across multiple different Azure offerings. Azure simplifies the process of creating the SAN by handling the deployment, management, and initial configuration.

Elastic SANs can provide up to millions of IOPS with double-digit GiB/s of throughput and single-digit-millisecond latencies.[46] Its benefit in a PostgreSQL use case would be if you had to store and centrally manage many terabytes of database data across multiple PostgreSQL instances. A tradeoff, however, would be that storage latencies on Elastic SAN would likely be higher than those of Premium SSD v2 instances or Azure Ultra Disks.

The individual volumes that you provision from the SAN share the SAN's performance values, up to a limit of 80,000 IOPS per volume. Individual volumes can have high IOPS and throughput, but the total IOPS and throughput of all your volumes can't exceed the total IOPS and throughput your SAN has. In addition to these performance benefits, Azure Elastic SAN brings a centralized approach to management and volume creation and maintenance, like in more traditional on-premises storage array approaches. You should consider Azure Elastic SAN as your storage solution when you need highly scalable storage with centralized management for your large-scale applications or when your workloads require low latency, high IOPS, and high throughput across multiple volumes.[47]

Because you can connect Azure Elastic SAN with your Azure Virtual Machines, you could choose to back your PostgreSQL databases with Azure Elastic SAN storage if your organization required that approach.

## Deploying PostgreSQL on Azure IaaS

### Creating an Ubuntu VM

Deploying PostgreSQL on an Azure VM involves creating the necessary VMs and configuring Azure specifics—such as selecting subscriptions, regions, and disks—and then accessing the VM's operating system and performing the necessary PostgreSQL install steps.

To demonstrate this, we used Ubuntu version 24.04 and a standard E2ads_v6 VM. We created an Ubuntu virtual machine by logging into the Azure Portal, navigating to the Virtual Machines service, and clicking Add. We set the Azure options and chose our OS image, Ubuntu 24.04 LTS. Next, we configured access essentials such as generating a new key pair and opening port 22. Finally, we configured disk and networking options, reviewed our VM, and clicked Create.

For step-by-step instructions, see below.

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. To open the Create a Virtual Machine wizard, click Create → Azure Virtual Machine.
3. On the Basics tab, set the following:
    a. Choose your Subscription from the dropdown menu.
    b. Choose your Resource group from the dropdown menu.
    c. Name the Virtual Machine.
    d. Choose your Region from the drop-down menu. (We used East US.)
    e. Under Availability options, choose Availability zone.
    f. Choose your Availability zone. (We used Zone 3.)
    g. From the Image drop-down menu, choose Ubuntu 24.04 LTS.
    h. Under VM architecture, select x64.
    i. Leave Azure Spot VM set to No.
    j. Select the VM size you wish to use. (We used Standard E2ads_v6.)
    k. Keep the Authentication type set to SSH public key.
    l. Choose a Username and Key pair name, and select Generate a new key pair.
    m. Leave Public inbound ports set to Allow selected ports.
    n. For Select inbound ports, choose SSH (22).
4. Click Next: Disks.
5. On the Disks tab, set the following:
    a. For the OS disk size, select 30GB.
    b. For the OS disk type, choose Premium SSD from the drop-down menu.
    c. Leave the default Encryption type.
6. Click Next: Networking.
7. On the Networking tab, leave all settings as default, and ensure the checkbox for accelerated networking is enabled.
8. Click Next: Management. On the Management tab, leave all defaults.

9. On the Advanced tab, leave all defaults. On the Tags tab, add any tags you wish to use.
10. On the Review + create tab, review your settings, and click Create.

After Azure has successfully deployed the VM, you can check its status in the Azure portal (see Figure 6).

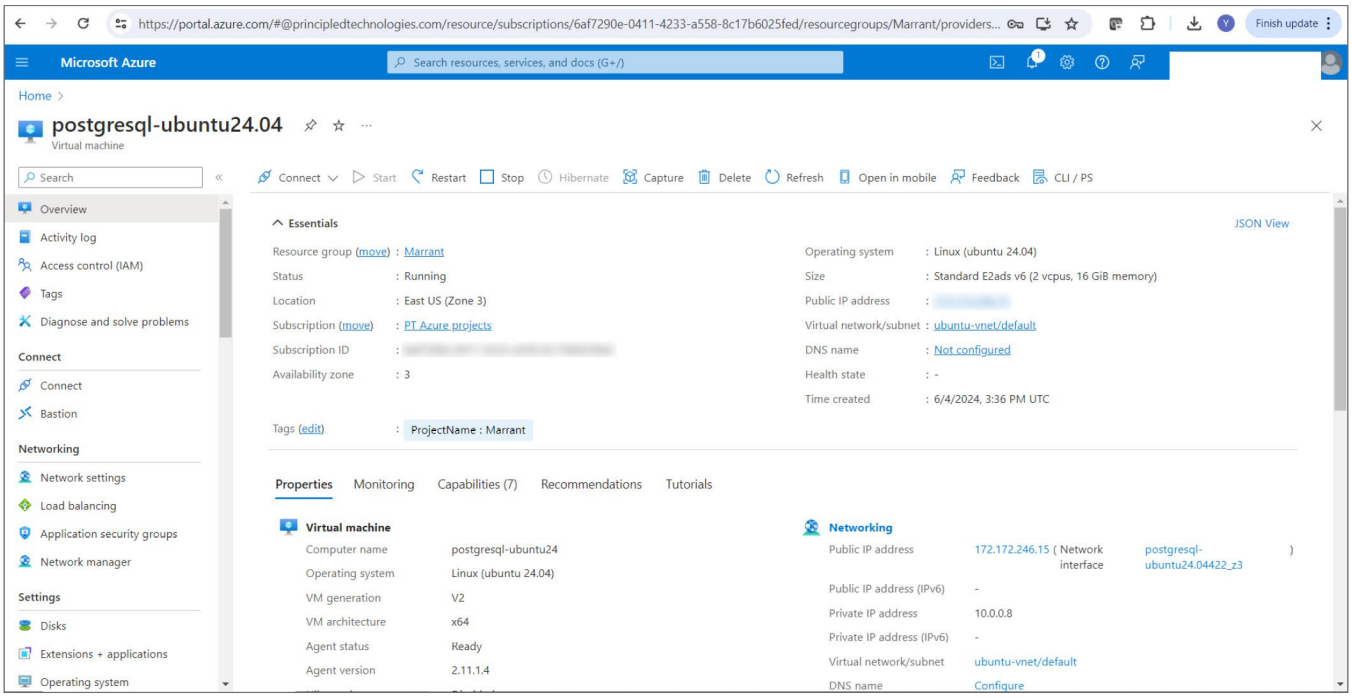



Figure 6: Screenshot of the Azure portal. Source: Principled Technologies.

## Installing PostgreSQL 16 on the Ubuntu VM

In this section, we demonstrate the steps to install PostgreSQL on the VM. Note: Consider using Ubuntu Pro (https://ubuntu.com/pro), which includes Expanded Security Maintenance (ESM), security updates, and other tools.

1. Use SSH to connect to the Ubuntu VM.
2. Install PostgreSQL 16:

```
sudo apt install postgresql-16
```

3. Connect to PostgreSQL and list the databases:

```
sudo -u postgres psql
```

### Compute considerations

#### VM sizing

VM sizing for PostgreSQL involves optimizing Azure resources such as vCPUs, memory, and storage to improve database performance, scalability, and reliability. Below, we look at some key components in a VM and present how they could impact performance.

#### CPU

PostgreSQL can benefit from multi-vCPU VMs on Azure for parallel query processing. You need to monitor CPU usage closely to make sure your PostgreSQL server works efficiently for complex operations such as joins, sorting, and aggregation. You can use the PostgreSQL view pg_stat_activity to begin troubleshooting expensive queries, where the view will show server process information such as process ID, connected user, and client name. Additionally, pgwatch is an open-source tool that can monitor CPU usage of your PostgreSQL database server. It can report which processes are consuming most of the CPU workload and potentially where you can make optimizations.[48]

#### Memory

Memory is crucial for databases to function properly. During a typical database query, if the data is not already in memory, the database retrieves it from disk and then temporarily stores it in memory. Then, the database takes further query actions, such as any aggregation, grouping, and sorting. This makes it important that your VM has enough memory.[49]

PostgreSQL performance benefits from caching data and indexes in memory. There are two layers of caching: the PostgreSQL memory cache, which the database directly manages and is the primary cache level where it holds data and indexes, and the operating system cache. While PostgreSQL may not directly manage the cached data in the OS memory, leveraging this layer of caching can greatly enhance read performance by storing data in the memory.

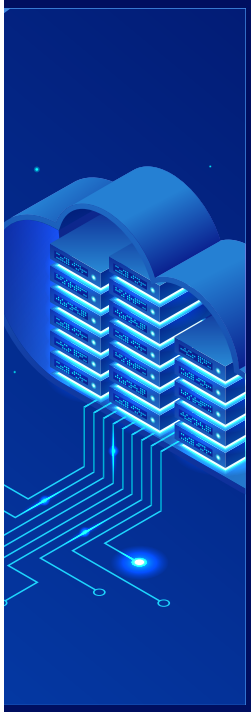
### Storage considerations

Choose storage that meets the performance requirements of your application. Organizations should first measure the application demands, then use those IOPS measurements and projections to size and select Azure disks. Based on those projections, customers can choose both the disk layout (i.e., using multiple disks), and the disk model. A good starting point can be to use Premium SSD v2 due to its flexibility with adjusting IO performance. If you do select Premium SSD (non v2), then you could consider striping multiple less expensive disks if you require more IOPS than a single larger more expensive disk could deliver.

When choosing Azure Disk Storage for your high-performance applications, it is important to consider the disk's IOPS, throughput, and latency characteristics to ensure they meet your application's needs. OLTP applications need to process many concurrent user requests immediately. These requests are typically small and random IO operations that require very high disk IOPS. On the other hand, OLAP or data warehouse applications need to scan and read large volumes of data. These operations are generally large and sequential IO tasks that require high disk throughput.[50]

You can also consider configuring multiple disks into a RAID group to combine their IOPS, throughput, and storage capacity to deliver even higher performance for your applications. On Linux, you can use the MDADM utility to stripe disks together and create a software RAID.[51]

Choose an appropriate file system type (e.g., ext4 or XFS) and mount option (e.g., noatime, nobarrier, or nodiratime) for PostgreSQL data directories to optimize disk IO performance and reduce file system overhead. Consider using separate file systems or mount points for PostgreSQL data, WAL logs, and temporary files to minimize contention and improve IO throughput (see the "Configuring multiple volumes" section below).



In addition to the IOPS and throughput limits on individual managed disks, users must also consider the performance limits associated with each VM itself. Each Azure VM has its own defined limit for IOPS and throughput, which, combined with the disk-specific limits, define the overall performance limits for the solution. Many VM series include limits for both cached and uncached disk performance, and burstable VMs have a specific limit for performance while bursting.

For example, say you were using a Standard_E16ads_v5 VM with an attached 256GiB Ultra Disk with 60,000 IOPS and 10,000 MB/s of provisioned throughput. Even though Azure has designed the disk to provide the given specifications, it will throttle the storage performance due to the inherent limits of the VM itself, which supports a maximum uncached IOPS of 25,600 and a maximum uncached throughput of 600 MB/s. Therefore, users must take both sets of limits into account, as well as the disk caching and bursting abilities of both the managed disks and the VMs. To find the storage performance limits of a particular VM or VM series, users can navigate to the "Sizes for virtual machines in Azure"[52] page, and select a VM type and VM series. Each series has an associated table listing the storage performance limits, which users can leverage to help plan their deployments.

### Configuring multiple volumes

Historically, in specific cases, database architects and storage architects have recommended configuring multiple volumes with multiple underlying disks for database applications. This is not always necessary, and you should evaluate your situation as to whether this approach would provide performance or reliability advantages.

Examples of when to do this may include:

- **Improved performance by separating data and logs:** The IO profile that data volumes ingest (small block, random, read/write) is likely different from the IO profile of a log volume (potentially larger block, sequential, write only).
- **Better scalability:** To increase the throughput available to you, you could stripe virtual disks together using the lvcreate tool to increase the IOPS capacities that are available to you in Azure on a single volume. Note that you should keep your organization's data redundancy policies in mind.

- **Streamlined backup and restore:** PostgreSQL DBMS might benefit from separating data and log volumes when it comes to backup and restore speeds.
- **Different disk types for data and logs:** Because Azure offers a variety of disk choices, you can select the disk type that suits your performance needs specific to both the data and log storage of your PostgreSQL database.

## OS considerations

### OS options

Adjust kernel parameters (sysctl settings) related to memory management, networking, disk IO, and process scheduling to optimize system performance for PostgreSQL. Common kernel parameters to tune include vm.swappiness, vm.dirty_ratio, vm.dirty_background_ratio, fs.file-max, net.core.somaxconn, and kernel.shmmax. For a detailed explanation, please refer to https://www.percona.com/blog/tune-linux-kernel-parameters-for-postgresql-optimization/.

## Database considerations

### PostgreSQL parameters

PostgreSQL has many parameters to optimize its performance. For an exhaustive list of all server parameters please visit https://www.postgresql.org/docs/current/runtime-config.html. We discuss some important ones in this section.

### Connection configuration parameters

Table 3: Connection configuration parameters. Source: Principled Technologies.

| Parameter | Description |
|---|---|
| max_connections | Sets the maximum number of concurrent connections to the PostgreSQL server. Setting an appropriate max_connections parameter ensures sufficient resources for active connections and prevents server overload. |

### Memory configuration parameters

Table 4: Memory configuration parameters. Source: Principled Technologies.

| Parameter | Description |
|---|---|
| shared_buffers | Defines the amount of memory the database allocates for caching data in shared memory. Increasing this parameter can improve read performance by caching more data in the memory and reducing disk read latency for frequently accessed data. |
| work_mem | Determines the amount of memory the database allocates for each internal sort or hash operation for every query execution. Increasing work_mem can improve performance for complex queries that involve sorting or hashing large datasets. However, please keep in mind that this parameter is per query, so if you have multiple sessions doing complex queries concurrently, the actual memory consumption will be multiple times of work_mem. |
| maintenance_work_mem | Specifies the amount of memory the database allocates for maintenance operations such as index creation, VACUUM, and ANALYZE. Increasing maintenance_work_mem can speed maintenance operations. |

### Write-ahead log configuration parameters

WAL is an important mechanism in PostgreSQL where the database first writes changes to a log before applying them to the actual data files. It keeps a record of all modifications to the database. By writing changes to the WAL before committing them to disk, PostgreSQL makes sure that it can replay transactions in case of a system crash or failure, maintaining data integrity and enabling efficient crash recovery. Below are several configuration parameters that can help improve WAL checkpoint performance.

Table 5: Write-ahead log configuration parameters. Source: Principled Technologies.

| Parameter | Description |
|---|---|
| wal_buffers | Sets the amount of memory the database allocates to buffer write-ahead logging data before it can flush it to disk. Increasing wal_buffers can improve write performance by reducing WAL write latency. |
| max_wal_size, min_wal_size | Sets the maximum and minimum size of the WAL log files, respectively. Modifying max_wal_size can affect how often checkpoints can occur and the amount of WAL data the database preserves in memory, and consequently affect write performance. Setting an optimal min_wal_size ensures enough WAL buffer capacity for write activities and can therefore let you avoid frequent checkpoints. |
| checkpoint_timeout | Defines the time duration between automatic WAL checkpoints. Adjusting checkpoint_timeout affects checkpoint frequency and consequently affects write performance. |
| commit_delay | Specifies a period that a WAL flush can wait before writing to disk. This delay can help reduce checkpoint overhead and improve performance by combining multiple small transactions into larger ones. |
| wal_compression | Can compress WAL data before writing to disk. It can help improve checkpoint performance by reducing the amount of data the database writes to disk, but can also cost extra CPU cycles by compressing the WAL log and decompressing it during WAL replay. |

### Query performance tuning

Understanding how PostgreSQL executes queries is important for optimizing query performance. Several commands are available to assist in automatically optimizing query performance.

Table 6: Query performance tuning parameters. Source: Principled Technologies.

| Parameter | Description |
|---|---|
| Analyze | After PostgreSQL receives a query, that query undergoes transformation. The query goes to the planner, which devises an execution plan for it. The planner relies on the pg_statistics table to assess the most efficient execution strategies. However, if the statistics become outdated, the execution plan may not be accurate or optimal. Therefore, running the analyze command becomes necessary to refresh statistics regarding the table contents in your database.[53] |
| Explain | Following the analyze command, explain is the next step in PostgreSQL query performance optimization. It outputs the plan devised by the database using the statistics generated by analyze. This plan provides insights into how the query will execute, whether it will utilize an index scan or a table scan, and so on. With this information, you can optimize the query for improved performance or update the statistics by running analyze once more.[54] |

## Scale considerations

### Scaling up

As your business grows and you begin to see workloads increase and your resource utilizations approaching their limits, you might want to consider scaling up your PostgreSQL VM to increase resources such as CPU, memory, and storage to handle more work.[55] You can change your VM size by using either the Azure Portal or Azure CLI. Scaling up will require you to first shut down the VM, which causes downtime. You should plan accordingly to minimize the impact of server outage. Scaling up to larger VM sizes will also incur higher costs. Please visit the Azure Linux Virtual Machines pricing calculator at https://azure.microsoft.com/pricing/details/virtual-machines/linux/ and make sure you understand the cost implications.

### Scaling out

Another option to accommodate increased workloads is to scale out your PostgreSQL VM. You can use Azure Virtual Machine Scale Sets (VMSS) to deploy and manage a set of identical and auto-scaling VMs. VMSS can automatically adjust the number of VMs in the scale set based on a predefined rule by scaling out VMs when workload increases and scaling in VMs when workload decreases. You can create VMSS from the Azure Portal or using Azure CLI. You can set the maximum and minimum instance count and scaling rules based on CPU, memory, or disk usage.[56] For example, you can add rules to automatically spin up a new VM in the scale set when your CPU usage exceeds a certain threshold. The ideal use case for this would be if you needed separate, smaller, isolated PostgreSQL instances.

## HA considerations

Azure offers a variety of high availability solutions to protect your applications and data from data center outages and maintenance events.[57]

### Using managed disks

As we note in the Azure Disk Storage features section, Azure managed disks are replicated three times and offer 3 nines of availability (99.999 percent) and 11 nines of durability (99.999999999 percent).[58] Because PostgreSQL is an application that would typically run on a single VM, using the Ultra Disks, Premium SSD v2, or Premium SSD offerings from Azure would work well as these types are all managed.[59]

### Availability zones

Availability zones are physically separate locations within an Azure region. Each zone has its own power, cooling, and networking. Deploying your PostgreSQL VMs across multiple availability zones can enhance resiliency against data center failures.[60] If one zone is compromised, your replicated apps and data are still available in another zone. Availability zones offer uptime percentages with Azure SLA of 99.99 percent.[61]

Azure has designed scale sets to support large-scale services that require high availability, auto-scaling, and easy management of a group of VMs. You can deploy virtual machines in a VMSS into multiple availability zones or fault domains. If one of these VM instances has a problem, customers can continue to access your PostgreSQL application through one of the other VM instances with minimal interruption.[62]

### Availability sets

You can deploy your PostgreSQL VMs on Availability sets, which distribute VMs across multiple fault and update domains. An update domain is a group of VMs and underlying physical hardware that you can update or reboot at the same time. A fault domain is a group of VMs that share common power source and network switch. By spreading VMs across multiple fault and update domains, Azure ensures that your application remains available even if one domain is being updated or failed. Availability sets meet the 99.95% Azure SLA.[63]

Microsoft recommends that users who are not already using Availability sets for their HA needs deploy PostgreSQL VMs across availability zones using VMSS flexible orchestration mode. They recommend this approach because VMSS flexible orchestration mode offers higher Azure SLA, larger VM scale, and enhanced VM management capabilities.[64]

### PostgreSQL

PostgreSQL provides many features that help customers build applications, efficiently manage data, protect data integrity, and provide fault-tolerant and high availability environments.

In this section, we focus on some key concepts and methods related to PostgreSQL high availability and replication. For a full list of PostgreSQL features, please visit https://www.postgresql.org/about/featurematrix/.

You can configure PostgreSQL in an HA cluster using tools such as Percona,[65] Pacemaker,[66] Patroni,[67] and repmgr.[68] These tools manage failover scenarios and automatically promote a standby server to primary in case of primary server failure.

A PostgreSQL HA configuration can minimize service outages and ensure your PostgreSQL database remains accessible in the event of planned system maintenance and upgrades, hardware failures, or network issues. Its automated failover mechanisms redirect client requests to PostgreSQL servers if the primary one become unavailable. PostgreSQL also uses streaming replication to replicate data from the primary server to standby ones.

It is important to consider Azure infrastructure design when designing your PostgreSQL HA setup. Azure groups its resources into Regions, which contain zones. One could choose to deploy an HA configuration for PostgreSQL both in a same-zone configuration, where both primary and secondary VMs reside within the same Azure zone, or across zones.

If you choose to deploy within the same zone, be sure VMs are deployed using the Azure Availability Set features, which ensures VMs are place in separate fault domains.

If you chose to deploy across zones to ensure issues in one zone do not affect the uptime of your application, you would first configure the primary VM in Zone 1, then configure the secondary VM in zone 2, and then configure replication options.

### Replication

PostgreSQL implements several methods to replicate data between servers for purposes of HA, load balancing, and performance improvement. The two main replication methods are streaming replication and logical replication.

### Streaming replication

Streaming replication is a built-in PostgreSQL feature.[69] It enables the primary PostgreSQL server to continuously stream change (WAL logs) to one or more standby servers as it generates them. You can use standby servers for read-only queries or to facilitate failover operations, as we discuss above.

There are two modes of streaming replication: synchronous replication, which confirms that PostgreSQL has transferred all changes to at least one of the standby servers, and asynchronous replication, which commits transactions without waiting for standby confirmations.

### Logical replication

Logical replication is also an integrated feature in PostgreSQL.[70] Rather than replicating physical data blocks in streaming replication, logical replication replicates changes at the row level (inserts, deletes, updates, etc.) in the WAL logs. The primary server (or publisher) sends these logical changes to standby servers (or subscribers) where they are applied to corresponding tables to sync with the primary database. Logical replication could be a good scenario to use for migration scenarios that require online migration.

For more information, refer to https://www.postgresql.org/docs/current/runtime-config-replication.html.

## Security considerations

When you deploy your PostgreSQL server on Azure IaaS, you can leverage numerous security features provided by both Azure and PostgreSQL. These features can help protect your database data, applications, and infrastructure.

### Azure confidential computing

Azure confidential computing is a set of technologies and features in Azure that work to protect data at rest and data in transit. These technologies include a number of different Azure offerings, including confidential VMs such as ECadsv5/ECasv5 powered by AMD EPYC™ CPUs. Microsoft lists the following features of confidential VMs:[71]

- "Encryption for 'data in use,' including the processor state and the virtual machine's memory.
- Host attestation helps you verify the full health and compliance of the server before data processing begins.
- Hardware Security Module (HSM) can be attached to guard the keys of confidential VM disks, which the tenant exclusively owns.
- New UEFI boot architecture supporting the guest OS for enhanced security settings and capabilities.
- A dedicated virtual Trusted Platform Module (TPM) certifies the health of the VM, provides hardened key management, and supports use cases such as BitLocker."

Only certain families of VMs offer Azure confidential computing, so you should check Microsoft documentation to ensure your relevant VM family is included. The ECasv5, ECadsv5,[72] DCasv5, and DCadsv5[73] series VMs are included in Azure confidential computing.

### Identity Management

#### Azure Key Vault

Azure Key Vault is a cloud-based service provided by Azure that allows you to securely store and manage confidential information such as keys, secrets, and certificates. Azure stores the information in a secure and centralized vault and protects it from unauthorized access. Azure Key Vaults can be either software-protected or hardware-protected by HSMs. Microsoft documentation states that "software-protected keys, secrets, and certificates are safeguarded by Azure, using industry-standard algorithms and key lengths. For situations where you require added assurance, you can import or generate keys in HSMs that never leave the HSM boundary."[74] When deploying a PostgreSQL server in Azure, you can store your sensitive information, such as your passwords, certificates, and database connection strings, in a secure Azure key vault to enhance security for your database server and applications. Furthermore, when using clientcert=verify-full in your PostgreSQL configuration, you can use Azure Key Vault to store the SSL certificate. For more information on PostgreSQL certificate connection options, see PostgreSQL documentation.[75]

### Azure Storage

#### Disk security

Azure provides two important security features for its managed disks: encryption and Private Links. Two different types of encryption are available: server-side encryption and Azure Disk Encryption. Server-side encryption, which Azure enables by default for all managed disks and their associated snapshots and images, provides encryption at rest. The encrypting keys can be managed either by Azure (platform-managed keys) or by the user (customer-managed keys). Azure Disk Encryption uses OS-level features to encrypt OS and data disks used by Azure VMs, such as BitLocker for Windows-based VMs Or DM-Crypt for Linux-based VMs.[76,77] The process integrates with Azure Key Vault, allowing users to manage the disk encryption keys and secrets.

Private Links let you import or export a managed disk within a secure Azure virtual network without exposing traffic to the public internet. This service allows users to generate a time-bound Shared Access Signature (SAS) uniform resource identifier (URI) for managed disks that enables them to export the disk data to other regions.

### Azure Networking

#### Azure Virtual Network

Azure Virtual Network (vNET) allows you to create private, isolated, and secure network environments in the Azure cloud. Each vNET is logically isolated from other vNETs. You can define IP address ranges, subnets, and network security groups (NSGs) to control traffic flow and enforce security policies between your PostgreSQL server VMs and applications within vNETs. To connect with other vNETs, Virtual Network Peering establishes peering connections between vNETs to enable seamless communication and resource sharing. To provide outbound internet connectivity, Azure Network Address Translation (NAT) gateway can route VM traffic to the internet while keeping VMs and compute resources private. If you are deploying a containerized PostgreSQL environment, vNET provides a container network interface (CNI) plugin to enable communication and connectivity for your containers within your Azure Kubernetes Service cluster.[78]

### Azure DDoS Protection

"A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service, or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic."[79] It is crucial to implement proactive security measures to protect against and mitigate the impact of DDoS attacks. Azure DDoS Protection automatically detects and mitigates even the most complex DDoS attacks against Azure resources, including virtual networks, public IP addresses, and Azure Load Balancers. It provides always-on monitoring and automatic mitigation capabilities to defend against both the network layer and the application layer attacks.[80]

### Azure Firewall

Azure Firewall allows you to define network security policies and rules to control inbound and outbound traffic for your Azure virtual network resources, based on IP addresses, ports, protocols, etc. You can create allow/deny rules, application rules, network rules, and NAT rules. Azure Firewall integrates with Azure Monitor for viewing and analyzing firewall logs.[81]

### Azure Load Balancer

Azure Load Balancer distributes inbound network traffic across backend pool VMs such as VMs and VM scale sets, based on configured load-balancing rules and health probes. It helps evenly distribute network workload, prevent bottlenecks, and optimize network resource utilizations. Azure Load Balancing supports both public load balancing and internal load balancing. A public load balancer can provide a public IP address for virtual machines inside your vNET. Public load balancers load balance internet traffic to your VMs. An internal or private load balancer handles traffic inside a virtual network or across vNET peering connections.[82]

## PostgreSQL security features and considerations

### Role-based access control (RBAC)

PostgreSQL RBAC enables you to grant specific privileges to roles and manage user permissions effectively. In PostgreSQL, roles act as users or groups of users, owning database objects such as tables and functions. Roles can manage permissions and access controls, and assign privileges on these objects to other roles.[83] Privileges in PostgreSQL include SELECT, INSERT, UPDATE, DELETE, TRUNCATE, and CREATE. Privileges allow granular control over database objects such as tables, views, schemas, and functions. You can grant or revoke privileges to roles or individual users by specifying what actions they are allowed or disallowed to perform on specific database objects.[84]

The GRANT and REVOKE commands in PostgreSQL allow administrators to manage access control by granting or revoking privileges on database objects (tables, views, functions, etc.) to users or roles. These commands help enforce security policies and maintain data integrity within PostgreSQL databases.[85]

### Row-level security

PostgreSQL offers row-level security (RLS), enabling tables to have row security policies that restrict access based on user-specific criteria. RLS policies control which rows users can query or modify, providing an additional layer of security beyond the standard SQL privilege system.[86]

### Authentication methods

PostgreSQL supports various authentication methods to verify the identity of users connecting to the database. The most common is the password authentication method. You can use the password command to send passwords in clear text. This method is straightforward, but PostgreSQL stores your passwords in the database without hashing, which makes them vulnerable to password sniffing attacks, and thus is not a best practice. You could also use the Message Digest Method 5 (MD5), where PostgreSQL stores passwords as MD5 hashes. During authentication, PostgreSQL hashes user passwords using MD5 and compares them with the stored hashes. The more secure method is Salted Challenge Response Authentication Mechanism (SCRAM), which uses SHA-256 hashing with a salted challenge-response mechanism.

In addition to password authentication, PostgreSQL supports other ways to authenticate users. For instance, the Generic Security Services Application Program Interface (GSSAPI) is often in Kerberos-based authentication setups. It enables single sign-on and centralized authentication mechanisms.[87] You can use a Lightweight Directory Access Protocol (LDAP) server to authenticate username and password pairs,[88] and PostgreSQL supports client certificate authentication, authenticating users via SSL and TLS certificates.[89] For a complete list of authentication methods, see https://www.postgresql.org/docs/current/auth-methods.html.

### Encryption considerations

For data security, consider all stages of the data lifecycle: data at rest on the cloud, in transit, and back to cloud storage. A good list of best practices could be:

- Always use disk encryption
- Encrypt data in transit
- Use SSL host certificates
- Consider targeted encryption at the row or cell level in PostgreSQL

### Across the network

PostgreSQL supports SSL encryption for PostgreSQL DB VMs. SSL encryption secures client-server communication and protects passwords, queries, and data in transit. "GSSAPI-encrypted connections encrypt all data sent across the network, including queries and data returned (no password is sent across the network)."[90]

In the pg_hba.conf file, administrators can define which hosts require SSL-encrypted connections (hostssl), ensuring enhanced security for sensitive data transmission.[91]

### Encryption for data at rest

PostgreSQL also provides mechanisms to protect very specific data at rest against unauthorized access, such as data that could be considered personally identifiable information (PII). Users can utilize the pgcrypto module for encrypting specific fields within database tables, helping to ensure data confidentiality. Additionally, PostgreSQL can perform data encryption at the file system or block level, preventing unauthorized access to unencrypted data if storage devices are compromised.[92]

### TCP/IP configuration

PostgreSQL uses TCP/IP as the default communication protocol for client-server connections. The PostgreSQL server listens for incoming TCP/IP connections on a specified port (default is 5432) and IP address.[93] The postgresql.conf file contains TCP/IP-related settings, including the listening address, port number, SSL/TLS configuration, and authentication methods.

### Firewall

Best practice is to deploy a firewall on the PostgreSQL database VM.[94] Linux distributions offer built-in firewalls such as ufw, firewalld, and iptables. A firewall enables you to set rules that govern allowed inbound and outbound traffic. You can define these rules based on parameters such as the local port (default is 5432 for PostgreSQL), protocol (IPv6 or TCP), and source address (often a list of subnets or specific addresses). Strictly controlling external access to the server is crucial for maintaining the security of your database.

### Network segmentation

Users can utilize network segmentation to separate PostgreSQL servers from other network traffic and restrict access to authorized users and applications. Within Azure, you can leverage Azure Virtual Network to create a private network specifically for your PostgreSQL server and associated applications.

## Backup considerations

There are several options for backing up your PostgreSQL VM on Azure. You can back it up at Azure level, back up the data directory from the file system level, or use PostgreSQL utilities to back up the database natively.

### Azure VM backup

When setting up an Azure VM for backup, a background backup process initiates based on your backup schedule.[95] To perform application or file system backups, you need to install a backup extension within the VM. The backup process starts by taking a snapshot of your VM and then transferring it to the vault. The first backup is a full backup, meaning Azure transfers all data blocks to the vault. In subsequent backups, Azure backs up only changed blocks.

Azure also supports encrypted VM backups using Storage Service Encryption (SSE) or Azure Disk Encryption. With SSE, Azure Storage automatically encrypts data at rest before storing it. Azure decrypts data during restore. Azure Disk Encryption encrypts both OS and data disks using either BitLocker encryption keys (BEKs) or Azure Key Vault key encryption (KEKs). Azure securely backs up both BEKs and KEKs and encrypts them within a key vault.[96]

### File system backup

Users can also use Linux utilities such as tar or rsync to make a copy of their data directory.[97] For example:

```
tar -cf backup.tar <path to data directory>
rsync -av <path to data directory> <path to backup location>
```

However, there are limitations with this method compared with the SQL-level backup. It requires shutting down the database first, before you make a copy of your database, and it backs up only the entire database VM, not individual tables and databases.

### PostgreSQL backup considerations

There are various approaches to backing up a PostgreSQL database: native PostgreSQL backups using pg_dump or pg_basebackup,[98] using backup managers such as pgBackRest and pgBarman[99,100] performing file-system-level backups, and continuous archiving with point-in-time restore.

### Using pg_dump

PostgreSQL provides the pg_dump utility to back up a database. It does not necessitate shutting down the database before taking the backup, and it ensures consistency even during concurrent usage. By default, pg_dump outputs to the standard output but can be directed to a text file.[101] The basic syntax is:

```
pg_dump dbname > dumpfile
```

Dump files are plain-text files containing the SQL commands required to rebuild the database to the saved state. To restore from such a script, you can use psql:

```
psql dbname < dumpfile
```

For more detailed information about PostgreSQL backup and restore, please visit https://www.postgresql.org/docs/current/backup.html.

### Continuous archiving

PostgreSQL maintains a write-ahead log in the pg_wal subdirectory under the data directory. The log records every change made to the database's data files. In case of a database system crash, PostgreSQL can replay the WAL logs to restore the database to its last consistency point. For continuous backup, you can simply continuously archive the WAL files. This is particularly valuable for large databases, where frequent full backups are impractical. You can restore the database to any point in time by replaying the WAL logs and stopping at the desired time.[102]

### Encrypting backups

You should ensure that your backup method of choice secures the data contained within it by way of encryption to protect against unauthorized access. There are a few options to encrypt your PostgreSQL backups.

### Encrypting with OS tools

You can use the pg_dump utility to perform logical backups of PostgreSQL databases.[103] To encrypt the backup file, you can use a combination of pg_dump and encryption tools provided by your operating system (i.e., gpg and ssl) or third-party software. To restore from an encrypted backup, you would first decrypt the backup file using the corresponding decryption command provided by your encryption tool, followed by using PostgreSQL tools such as pg_restore or psql to restore your database from backup.

### Encrypting backups with pgBackRest

pgBackRest is an open-source backup and restore tool for PostgreSQL.[104] You can configure encryption options in the pgBackRest configuration file. For example, you can generate one by running:

```
openssl rand -base64 48
```

You can then add your cipher type and cipher key to the pgBackRest configuration file at /etc/pgbackrest.conf.

Once you have configured encryption in the pgBackRest configuration file, you can use the pgbackrest command line tool to perform encrypted backups. pgBackRest supports three backup types: Full backup is a complete copy of your entire dataset, differential backup includes all data that has changed since the last full backup, and incremental backup backs up only the files that have changed since the last full or differential backup. To restore from an encrypted backup created with pgBackRest, you would use the pgBackRest restore command, specifying the backup set and target location for the restore operation. The restore process handles decryption of the backup files automatically based on the encryption settings configured in the pgbackrest.conf file.

For detailed information about how to configure and use pgBackRest to back up and restore your database, please visit https://pgbackrest.org/user-guide.html#quickstart/configure-encryption.

## Monitoring and maintenance

Monitoring any application is a key component of your organization's planning and strategy. In the context of deploying PostgreSQL to Microsoft Azure, you can use a variety of monitoring tools and techniques across cloud, OS, and database layers to optimize for your configuration.

### Azure Monitor

Azure Monitor is a cloud monitoring solution offered by Microsoft that collects telemetry data from multiple types of resources and gives the user the ability to analyze, extract, and use the data in various ways, including alerting.

According to Microsoft, "Azure Monitor can monitor these types of resources in Azure, other clouds, or on-premises:

- Applications
- Virtual machines
- Guest operating systems
- Containers including Prometheus metrics
- Databases
- Security events in combination with Azure Sentinel
- Networking events and health in combination with Network Watcher
- Custom sources that use the APIs to get data into Azure Monitor"[105]

Azure Monitor uses a variety of techniques to collect data, including logs and metrics collection, agent-based collection, and third-party sources.

On Azure Virtual Machines, you can achieve basic telemetry such as CPU utilization, logical disk IOPS, and available memory by simply enabling Azure Virtual Machine Insights. For additional metrics, you can install the Microsoft Azure Monitor agent. For more details on agent installation, see https://learn.microsoft.com/azure/azure-monitor/agents/azure-monitor-agent-manage?tabs=azure-cli#install.

## Monitoring at the PostgreSQL level

PostgreSQL offers a wide range of tools designed to assist users in monitoring performance, system health, and database activities, which helps ensure optimal performance and diagnose issue. For a comprehensive list of monitoring utilities, please refer to https://www.postgresql.org/docs/current/monitoring.html.

One of the primary tools that PostgreSQL natively offers is a series of views accessible through its Cumulative Statistics System. Each view, which you can query via the psql command, offers information and statistics on specific information within the PostgreSQL VM. Examples include:

- pg_stat_activity presents information related to the ongoing activity of each server process, with one row per process.
- pg_stat_database contains statistics at the database level, with one row for each database in the cluster and an additional row for shared objects.
- pg_stat_bgwrite provides global statistics for the background writer, including metrics such as the number of allocated buffers, buffers written directly by the background writer, buffers the system wrote during checkpoints, timestamps, etc.
- pg_stat_wal offers insights into write-ahead logging activity, such as the frequency of WAL files synced to disk via fsync requests, WAL data written to disk due to full WAL buffers, total WAL logs generated, etc.
- pg_stat_all_indexes reports statistics for each index in the current database, reporting metrics such as the number of index scans initiated, the timing of the last scan, returned index entries, etc.
- pg_statio_all_tables provides IO statistics for each table in the database, offering valuable insights into disk activity.
- pg_locks displays outstanding locks in the lock manager, helping administrators identify potential sources of contention among clients and assess the impact of locks on overall database performance.[106]

PostgreSQL also supports reporting the progress of specific commands during execution. For instance, pg_stat_progress_create_index monitors progress during index creation.

In addition to PostgreSQL's native tools, users can leverage Linux utilities—such as top, mpstat, iostat, and nmon—to monitor various VM aspects, such as CPU usage, memory utilization, disk IO, and network traffic.

## Other tools to consider

### Percona PostgreSQL

Percona PostgreSQL is a distribution of the PostgreSQL database.[107] It is based on the open-source PostgreSQL database and includes additional features, optimizations, and tools designed to enhance performance, scalability, and manageability for PostgreSQL deployments. Percona includes tools such as the following:

- Patroni is an HA solution for PostgreSQL. It orchestrates and manages PostgreSQL HA clusters, and orchestrates automated failover and replication.[108]
- pgAudit provides detailed session or object audit logging via the standard PostgreSQL logging facility.[109]
- PgBouncer is a lightweight connection pool for PostgreSQL databases. Acting as a middleware layer between client applications and PostgreSQL servers, PgBouncer manages and optimizes database connections to enhance performance and resource utilization.[110] Instead of creating a new connection for each client request, PgBouncer reuses existing connections from the pool. This approach can help reduce latency and overhead typically associated with establishing and closing connections to the PostgreSQL server. You can also configure PgBouncer to support failover capabilities, allowing it to route client connections to various PostgreSQL servers (primary or standby) based on load balancing or failover policies.

For a complete list of Percona features and tools, please visit https://docs.percona.com/postgresql/16/index.html.

### Charmed PostgreSQL

Charmed PostgreSQL is an enterprise-grade solution to secure and automate the deployment, maintenance, and upgrades of your PostgreSQL databases.[111] Users can deploy it on Azure VMs and AKS. Charmed PostgreSQL provides necessary features for deployments of PostgreSQL:

- Patroni for high availability with built-in replication and automatic failover[112]
- PgBouncer for connection pooling and client failovers[113]
- Read scaling on replicas for resource utilization[114]
- Endpoint Management for efficient read/write operations on the primary VM and scaling read operations on replicas[115]
- PgBackRest for backup and restore management[116]

Canonical also offers managed PostgreSQL services and consulting for tailored solutions, including deployment optimization, support, and training. For more information, please visit the Canonical PostgreSQL page.[117]

## Conclusion

Choosing the right deployment model for PostgreSQL on Azure depends on your organization's specific needs for control, flexibility, and management. Azure Database for PostgreSQL – Flexible Server offers a fully managed PaaS platform that can simplify operations. The PaaS solutions are a strong contender for applications requiring rapid deployment and scaling, minimal administrative overhead, and seamless scalability. Conversely, deploying PostgreSQL on Azure IaaS provides greater control over the infrastructure. This approach offers flexibility to configure the operating system, storage, and networking according to customer needs and could be ideal for legacy applications, complex regulatory environments, or scenarios requiring higher customization. This guide presented how to use both Azure approaches as well as several recommendations on AMD EPYC™ CPU-based infrastructure to consider. Whether you want the simplicity of PaaS or maximum control with IaaS, Azure provides the tools and means to deploy PostgreSQL databases effectively in the cloud.

---


1. PostgreSQL, "2. A Brief History of PostgreSQL," accessed May 15, 2024, https://www.postgresql.org/docs/current/history.html.
2. Azure, "Server parameters in Azure Database for PostgreSQL - Flexible Server," accessed October 4, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-server-parameters.
3. Azure, "Supported versions for Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-supported-versions.
4. Azure, "Scheduled maintenance in Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-maintenance.
5. Azure, "Azure AI extension on Azure Database for PostgreSQL – Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/generative-ai-azure-overview.
6. Microsoft, "Scandinavian Airlines speeds app development, lowers costs with Azure Database for PostgreSQL," accessed September 23, 2024, https://customers.microsoft.com/story/1431763554334987166-scandinavian-air-lines-speeds-app-development-lowers-costs-azure-database-postgresql.
7. Microsoft, "Allego scales for EV growth by migrating to Azure Database for PostgreSQL - Flexible Server," accessed September 23, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/generative-ai-azure-overview.
8. Azure, "What's happening to Azure Database for PostgreSQL - Single Server after the retirement announcement?" accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/single-server/whats-happening-to-post-gresql-single-server.
9. Azure, "Sizes for virtual machines in Azure," accessed September 9, 2024, https://learn.microsoft.com/azure/virtual-machines/sizes/overview.
10. Azure, "Compute options in Azure Database for PostgreSQL – Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-compute.

11. Azure, "B-series burstable virtual machine sizes," accessed September 6, 2024, https://learn.microsoft.com/azure/virtual-machines/sizes-b-series-burstable.
12. Azure, "Virtual Machine series," accessed September 6, 2024, https://azure.microsoft.com/pricing/details/virtual-machines/series/.
13. Sumit Jadhav, "Virtual machines in Azure," accessed September 9, 2024, https://medium.com/@sumitjadhav6067/virtual-machines-in-azure-d51408147f4f.
14. Azure, "Virtual Machine series."
15. Azure, "Virtual Machine series."
16. Azure, "Compute options in Azure Database for PostgreSQL - Flexible Server."
17. Microsoft, "Dasv5 and Dadsv5-series" accessed August 30, 2024, https://learn.microsoft.com/azure/virtual-machines/dasv5-dadsv5-series#dadsv5-series.
18. Microsoft, "Easv5 and Eadsv5-series," accessed August 30, 2024, https://learn.microsoft.com/azure/virtual-machines/easv5-eadsv5-series#eadsv5-series.
19. Azure, "Compute options in Azure Database for PostgreSQL - Flexible Server."
20. Azure, "Backup and restore in Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-backup-restore.
21. Azure, "High availability (Reliability) in Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/reliability/reliability-postgresql-flexible-server.
22. Azure, "High availability (Reliability) in Azure Database for PostgreSQL - Flexible Server."
23. Azure, "High availability (Reliability) in Azure Database for PostgreSQL - Flexible Server."
24. Azure, "High availability (Reliability) in Azure Database for PostgreSQL - Flexible Server."
25. Azure, "Data encryption with a customer managed key for Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024 https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-data-encryption.
26. Azure, "Scheduled maintenance in Azure Database for PostgreSQL - Flexible Server."
27. Azure, "Microsoft Entra authentication for Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-azure-ad-authentication.
28. Azure, "Private Network Access using virtual network integration for Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-networking-private.
29. Azure, "Backup and restore in Azure Database for PostgreSQL - Flexible Server," accessed September 10, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-backup-restore.
30. Azure, "Monitor metrics on Azure Database for PostgreSQL - Flexible Server," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/concepts-monitoring.
31. Azure, "Error logs in Azure Database for PostgreSQL - Flexible Server (Preview)," accessed September 9, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/how-to-configure-and-access-logs.
32. Azure, "Azure AI extension on Azure Database for PostgreSQL - Flexible Server," accessed September 19, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/generative-ai-azure-overview.
33. Azure, "How to enable and use pgvector on Azure Database for PostgreSQL - Flexible Server," accessed September 19, 2024, https://learn.microsoft.com/azure/postgresql/flexible-server/how-to-use-pgvector.
34. Divine Odazie, "Vector Similarity Search with PostgreSQL's pgvector – A Deep Dive," accessed September 19, 2024, https://severalnines.com/blog/vector-similarity-search-with-postgresqls-pgvector-a-deep-dive/.
35. Azure, "Compare support plans," accessed September 5, 2024, https://azure.microsoft.com/support/plans/.
36. Azure, "Azure Support: Standard," accessed September 9, 2024, https://azure.microsoft.com/support/plans/standard/.
37. Azure, "Azure Support: Professional Direct (ProDirect)," accessed September 9, 2024, https://azure.microsoft.com/support/plans/prodirect/.
38. Microsoft, "Easv6 and Eadsv6-series (Preview)," accessed May 15, 2024, https://learn.microsoft.com/azure/virtual-machines/easv6-eadsv6-series.
39. Microsoft, "Falsv6, Fasv6, and Famsv6-series (Preview)," accessed May 15, 2024, https://learn.microsoft.com/azure/virtual-machines/fasv6-falsv6-series.

40. PostgreSQL, "17.6. Supported Platforms," accessed May 15, 2024, https://www.postgresql.org/docs/current/supported-platforms.html.
41. Microsoft, "Introduction to Azure managed disks," accessed May 15, 2024, https://learn.microsoft.com/azure/virtual-machines/managed-disks-overview.
42. Microsoft, "Introduction to Azure managed disks."
43. Microsoft, "Overview of Azure Disk Backup," accessed May 15, 2024, https://learn.microsoft.com/azure/backup/disk-backup-overview.
44. Microsoft, "Create an incremental snapshot for managed disks," accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machines/disks-incremental-snapshots.
45. Microsoft, "What is Azure Elastic SAN?" accessed May 15, 2024, https://learn.microsoft.com/azure/storage/elastic-san/elastic-san-introduction.
46. Microsoft, "What is Azure Elastic SAN?"
47. Microsoft, "Azure Elastic SAN," accessed June 13, 2024, https://azure.microsoft.com/products/storage/elastic-san.
48. Timescale, "PostgreSQL Performance Tuning: How to Size Your Database," accessed May 15, 2024, https://www.timescale.com/learn/postgresql-performance-tuning-how-to-size-your-database.
49. Timescale, "PostgreSQL Performance Tuning: How to Size Your Database."
50. Microsoft, "Azure premium storage: Design for high performance," accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machines/premium-storage-performance.
51. Microsoft, "Configure Software RAID on Linux," accessed June 13, 2024, https://learn.microsoft.com/previous-versions/azure/virtual-machines/linux/configure-raid.
52. Microsoft, "Sizes for virtual machines in Azure," accessed May 15, 2024, https://learn.microsoft.com/azure/virtual-machines/sizes.
53. PostgreSQL, "ANALYZE," accessed May 15, 2024, https://www.postgresql.org/docs/16/sql-analyze.html.
54. PostgreSQL, "EXPLAIN," accessed May 15, 2024, https://www.postgresql.org/docs/16/sql-explain.html.
55. Microsoft, "Scale up an app in Azure App Service," accessed June 13, 2024, https://learn.microsoft.com/azure/app-service/manage-scale-up.
56. Microsoft, "What are Virtual Machine Scale Sets?" accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machine-scale-sets/overview.
57. Microsoft, "Availability sets overview," accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machines/availability-set-overview.
58. Microsoft, "Best practices for achieving high availability with Azure virtual machines and managed disks," accessed July 17, 2024, https://learn.microsoft.com/azure/virtual-machines/disks-high-availability.
59. Microsoft, "Best practices for achieving high availability with Azure virtual machines and managed disks," accessed July 17, 2024, https://learn.microsoft.com/azure/virtual-machines/disks-high-availability.
60. Microsoft, "What are availability zones?" accessed June 13, 2024, https://learn.microsoft.com/azure/reliability/availability-zones-overview.
61. Microsoft, "What are Virtual Machine Scale Sets?" accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machine-scale-sets/overview.
62. Microsoft, "What are Virtual Machine Scale Sets?" accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machine-scale-sets/overview.
63. Microsoft, "Availability sets overview," accessed June 13, 2024, https://learn.microsoft.com/azure/virtual-machines/availability-set-overview.
64. Microsoft, "Best practices for achieving high availability with Azure virtual machines and managed disks," accessed July 17, 2024, https://learn.microsoft.com/azure/virtual-machines/disks-high-availability.
65. Pete Scott, "Setting Up and Deploying PostgreSQL for High Availability," accessed May 15, 2024, https://www.percona.com/blog/setting-up-and-deploying-postgresql-for-high-availability/.
66. Naresh Movva, "PostgreSQL — Active/Passive HA Cluster with Pacemaker," accessed May 15, 2024, https://medium.com/@naresh.mdm/postgresql-active-passive-ha-cluster-with-pacemaker-eccfda3e0832.

67. Patroni, "Introduction," accessed May 15, 2024, https://patroni.readthedocs.io/en/latest/.
68. repmgr, "Replication Manager for PostgreSQL Clusters," accessed May 15, 2024, https://www.repmgr.org/.
69. PostgreSQL, "27.2.5. Streaming Replication," accessed May 15, 2024, https://www.postgresql.org/docs/current/warm-standby.html#STREAMING-REPLICATION.
70. PostgreSQL, "Chapter 31. Logical Replication," accessed May 15, 2024, https://www.postgresql.org/docs/current/logical-replication.html.
71. Microsoft, "Azure confidential virtual machines FAQ," accessed July 17, 2024, https://learn.microsoft.com/azure/confidential-computing/confidential-vm-faq.
72. Microsoft, "ECasv5 and ECadsv5-series," accessed July 17, 2024, https://learn.microsoft.com/azure/virtual-machines/ecasv5-ecadsv5-series.
73. Microsoft, "DCasv5 and DCadsv5-series confidential VMs," accessed July 17, 2024, https://learn.microsoft.com/azure/virtual-machines/dcasv5-dcadsv5-series.
74. Microsoft, "About Azure Key Vault," accessed May 15, 2024, https://learn.microsoft.com/azure/key-vault/general/overview.
75. PostgreSQL, "19.9. Secure TCP/IP Connections with SSL," accessed June 27, 2024, https://www.postgresql.org/docs/current/ssl-tcp.html.
76. Microsoft, "Azure Disk Encryption for Windows VMs," accessed May 15, 2024, https://learn.microsoft.com/azure/virtual-machines/windows/disk-encryption-overview.
77. Microsoft, "Azure Disk Encryption for Linux VMs," accessed May 15, 2024, https://learn.microsoft.com/azure/virtual-machines/linux/disk-encryption-overview.
78. Microsoft, "Azure Virtual Network," accessed May 15, 2024, https://azure.microsoft.com/products/virtual-network/.
79. Cloudflare, "What is a DDoS attack?" accessed May 15, 2024, https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/.
80. Microsoft, "Azure DDoS Protection," accessed May 15, 2024, https://azure.microsoft.com/products/ddos-protection.
81. Microsoft, "Azure Firewall," accessed May 15, 2024, https://azure.microsoft.com/products/azure-firewall/.
82. Microsoft, "What is Azure Load Balancer?" accessed May 15, 2024, https://learn.microsoft.com/azure/load-balancer/load-balancer-overview.
83. PostgreSQL, "Chapter 22. Database Roles," accessed May 15, 2024, https://www.postgresql.org/docs/current/user-manag.html.
84. PostgreSQL, "5.7. Privileges," accessed May 15, 2024, https://www.postgresql.org/docs/current/ddl-priv.html.
85. PostgreSQL, "GRANT," accessed May 15, 2024, https://www.postgresql.org/docs/current/sql-grant.html.
86. PostgreSQL, "5.8. Row Security Policies," accessed May 15, 2024, https://www.postgresql.org/docs/current/ddl-rowsecurity.html.
87. PostgreSQL, "21.6. GSSAPI Authentication," accessed May 15, 2024, https://www.postgresql.org/docs/current/gssapi-auth.html.
88. PostgreSQL, "21.10. LDAP Authentication," accessed May 15, 2024, https://www.postgresql.org/docs/current/auth-ldap.html.
89. PostgreSQL, "21.12. Certificate Authentication," accessed May 15, 2024, https://www.postgresql.org/docs/current/auth-cert.html.
90. PostgreSQL, "19.8. Encryption Options," accessed May 15, 2024, https://www.postgresql.org/docs/current/encryption-options.html.
91. PostgreSQL, "9.9. Secure TCP/IP Connections with SSL," accessed May 15, 2024, https://www.postgresql.org/docs/current/ssl-tcp.html.
92. PostgreSQL, "19.8. Encryption Options," accessed May 15, 2024, https://www.postgresql.org/docs/current/encryption-options.html.
93. PostgreSQL, "20.3 Connections and Authentication," accessed May 15, 2024, https://www.postgresql.org/docs/current/runtime-config-connection.html.

94. Winnie Ondara, "How to Secure Your PostgreSQL Server?" accessed May 15, 2024, https://www.cherryservers.com/blog/how-to-secure-your-postgresql-server.
95. Microsoft, "An overview of Azure VM backup," accessed May 15, 2024, https://learn.microsoft.com/azure/backup/backup-azure-vms-introduction.
96. Microsoft, "An overview of Azure VM backup."
97. PostgreSQL, "26.2. File System Level Backup," accessed May 15, 2024, https://www.postgresql.org/docs/current/backup-file.html.
98. PostgreSQL, "pg_basebackup," accessed June 13, 2024, https://www.postgresql.org/docs/current/app-pgbasebackup.html.
99. PostgreSQL, "pgBackRest 2.46 Released," accessed June 13, 2024, https://www.postgresql.org/about/news/pgbackrest-246-released-2655/.
100. Barman, "Barman," accessed June 13, 2024, https://pgbarman.org/.
101. PostgreSQL, "pg_dump," accessed May 15, 2024, https://www.postgresql.org/docs/current/app-pgdump.html.
102. PostgreSQL, "26.3. Continuous Archiving and Point-in-Time Recovery (PITR)," accessed May 15, 2024, https://www.postgresql.org/docs/current/continuous-archiving.html.
103. LinkedIn, "How can you encrypt database backups for maximum security?" accessed May 15, 2024, https://www.linkedin.com/advice/3/how-can-you-encrypt-database-backups-maximum-security-bgvqc.
104. pgBackRest, "pgBackRest User Guide," accessed May 15, 2024, https://pgbackrest.org/user-guide.html.
105. Microsoft, "Azure Monitor overview," accessed May 15, 2024, https://learn.microsoft.com/azure/azure-monitor/overview.
106. PostgreSQL, "54.12. pg_locks," accessed May 15, 2024, https://www.postgresql.org/docs/current/view-pg-locks.html.
107. Percona, "Percona Distribution for PostgreSQL," accessed May 15, 2024, https://www.percona.com/postgresql/software/postgresql-distribution.
108. Patroni, "Introduction," accessed May 15, 2024, https://patroni.readthedocs.io/en/latest/.
109. GitHub, "pgaudit," accessed May 15, 2024, https://github.com/pgaudit/pgaudit.
110. PgBouncer, "pgbouncer," accessed May 15, 2024, https://www.pgbouncer.org/usage.html.
111. Canonical, "Charmed PostgreSQL VM Tutorial," accessed June 13, 2024, https://canonical.com/data/docs/postgresql/iaas/t-overview.
112. Canonical Juju, "Configuring Patroni-based PostgreSQL asynchronous replication," accessed June 13, 2024, https://discourse.charmhub.io/t/deprecated-charmed-postgresql-k8s-how-to-patroni-async-replication/12226.
113. Canonical Juju,"PgBouncer," accessed June 13, 2024, https://charmhub.io/pgbouncer.
114. Canonical Juju, "Charmed PostgreSQL How-To | Scale units," accessed June 13, 2024, https://discourse.charmhub.io/t/charmed-postgresql-how-to-scale-units/9689.
115. Canonical, "Scale your replicas," accessed June 13, 2024, https://canonical.com/data/docs/postgresql/k8s/t-scale.
116. Canonical Juju, "Charmed PostgreSQL VM," accessed June 13, 2024, https://charmhub.io/postgresql/actions.
117. Canonical, "PostgreSQL operations, simplified," accessed June 13, 2024, https://canonical.com/data/postgresql.

# Appendix

## Sample ARM template for deploying Azure Database for PostgreSQL

**template.json**

```
{
    "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/
deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "administratorLogin": {
            "type": "string"
        },
        "administratorLoginPassword": {
            "type": "securestring"
        },
        "location": {
            "type": "string"
        },
        "serverName": {
            "type": "string"
        },
        "serverEdition": {
            "type": "string"
        },
        "storageSizeGB": {
            "type": "int"
        },
        "haEnabled": {
            "type": "string",
            "defaultValue": "Disabled"
        },
        "availabilityZone": {
            "type": "string",
            "defaultValue": ""
        },
        "standbyAvailabilityZone": {
            "type": "string",
            "defaultValue": ""
        },
        "version": {
            "type": "string"
        },
        "tags": {
            "type": "object",
            "defaultValue": {}
```

```
        },
        "firewallRules": {
            "type": "object",
            "defaultValue": {}
        },
        "storageAutogrow": {
            "type": "string",
            "defaultValue": "Disabled"
        },
        "backupRetentionDays": {
            "type": "int"
        },
        "geoRedundantBackup": {
            "type": "string"
        },
        "vmName": {
            "type": "string",
            "defaultValue": "Standard_D4s_v3"
        },
        "identityData": {
            "type": "object",
            "defaultValue": {}
        },
        "dataEncryptionData": {
            "type": "object",
            "defaultValue": {}
        },
        "apiVersion": {
            "type": "string",
            "defaultValue": "2022-12-01"
        },
        "aadEnabled": {
            "type": "bool",
            "defaultValue": false
        },
        "aadData": {
            "type": "object",
            "defaultValue": {}
        },
        "authConfig": {
            "type": "object",
            "defaultValue": {}
        },
        "network": {
            "type": "object",
            "defaultValue": {}
```

```
        },
        "iopsTier": {
            "type": "string",
            "defaultValue": ""
        },
        "storageIops": {
            "type": "int",
            "defaultValue": 0
        },
        "throughput": {
            "type": "int",
            "defaultValue": 0
        },
        "storageType": {
            "type": "string",
            "defaultValue": ""
        },
        "guid": {
            "type": "string",
            "defaultValue": "[newGuid()]"
        }
    },
    "variables": {
        "firewallRules": "[parameters('firewallRules').rules]"
    },
    "resources": [
        {
            "apiVersion": "[parameters('apiVersion')]",
            "location": "[parameters('location')]",
            "name": "[parameters('serverName')]",
            "identity": "[if(empty(parameters('identityData')), json('null'),
parameters('identityData'))]",
            "properties": {
                "createMode": "Default",
                "version": "[parameters('version')]",
                "administratorLogin": "[parameters('administratorLogin')]",
                "administratorLoginPassword": "[parameters('administratorLogi
nPassword')]",
                "Network": "[if(empty(parameters('network')), json('null'),
parameters('network'))]",
                "availabilityZone": "[parameters('availabilityZone')]",
                "Storage": {
                    "StorageSizeGB": "[parameters('storageSizeGB')]",
                    "Type": "[if(empty(parameters('storageType')), json('null'
),parameters('storageType'))]",
                    "Autogrow": "[parameters('storageAutogrow')]",
```

```
                    "tier": "[if(empty(parameters('iopsTier')), json('null'),
parameters('iopsTier'))]",
                    "Iops": "[if(equals(parameters('storageIops'), 0),
json('null'), parameters('storageIops'))]",
                    "Throughput": "[if(equals(parameters('throughput'), 0),
json('null'), parameters('throughput'))]"
                },
                "Backup": {
                    "backupRetentionDays": "[parameters('backupRete
ntionDays')]",
                    "geoRedundantBackup": "[parameters('geoRedundantBackup')]"
                },
                "highAvailability": {
                    "mode": "[parameters('haEnabled')]",
                    "standbyAvailabilityZone": "[parameters('standbyAvaila
bilityZone')]"
                },
                "dataencryption": "[if(empty(parameters('dataEncryptionDa
ta')), json('null'), parameters('dataEncryptionData'))]",
                "authConfig": "[if(empty(parameters('authConfig')),
json('null'), parameters('authConfig'))]"
            },
            "sku": {
                "name": "[parameters('vmName')]",
                "tier": "[parameters('serverEdition')]"
            },
            "tags": "[parameters('tags')]",
            "type": "Microsoft.DBforPostgreSQL/flexibleServers"
        },
        {
            "condition": "[parameters('aadEnabled')]",
            "type": "Microsoft.Resources/deployments",
            "apiVersion": "2018-05-01",
            "name": "[concat('addAdmins-', parameters('guid'))]",
            "dependsOn": [
                "[concat('Microsoft.DBforPostgreSQL/flexibleServers/',
parameters('serverName'))]"
            ],
            "properties": {
                "mode": "Incremental",
                "template": {
                    "$schema": "http://schema.management.azure.com/
schemas/2015-01-01/deploymentTemplate.json#",
                    "contentVersion": "1.0.0.0",
                    "resources": [
                        {
```

```
                            "type": "Microsoft.DBforPostgreSQL/flexibleServers/
administrators",
                            "name": "[concat(parameters('serverName'),'/',
parameters('aadData').objectId)]",
                            "apiVersion": "[parameters('apiVersion')]",
                            "properties": {
                                "tenantId":
"[parameters('aadData').tenantId]",
                                "principalName": "[parameters('aadData').
principalName]",
                                "principalType": "[parameters('aadData').
principalType]"
                            }
                        }
                    ]
                }
            }
        },
        {
            "condition": "[greater(length(variables('firewallRules')), 0)]",
            "type": "Microsoft.Resources/deployments",
            "apiVersion": "2019-08-01",
            "name": "[concat('firewallRules-', parameters('guid'), '-',
copyIndex())]",
            "copy": {
                "count": "[if(greater(length(variables('firewallRules')), 0),
length(variables('firewallRules')), 1)]",
                "mode": "Serial",
                "name": "firewallRulesIterator"
            },
            "dependsOn": [
                "[concat('Microsoft.DBforPostgreSQL/flexibleServers/',
parameters('serverName'))]",
                "[concat('Microsoft.Resources/deployments/addAdmins-',
parameters('guid'))]"
            ],
            "properties": {
                "mode": "Incremental",
                "template": {
                    "$schema": "http://schema.management.azure.com/
schemas/2014-04-01-preview/deploymentTemplate.json#",
                    "contentVersion": "1.0.0.0",
                    "resources": [
                        {
                            "type": "Microsoft.DBforPostgreSQL/flexibleServers/
firewallRules",
```

```
                            "name": "[concat(parameters('serverName'),'/',vari
ables('firewallRules')[copyIndex()].name)]",
                            "apiVersion": "[parameters('apiVersion')]",
                            "properties": {
                                "StartIpAddress": "[variables('firewallRules')
[copyIndex()].startIPAddress]",
                                "EndIpAddress": "[variables('firewallRules')
[copyIndex()].endIPAddress]"
                            }
                        }
                    ]
                }
            }
        ]
}
```

**parameters.json**

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "serverName": {
            "value": "pt-postgresql-demo"
        },
        "location": {
            "value": "eastus"
        },
        "serverEdition": {
            "value": "GeneralPurpose"
        },
        "storageSizeGB": {
            "value": 128
        },
        "haEnabled": {
            "value": "ZoneRedundant"
        },
        "backupRetentionDays": {
            "value": 14
        },
        "geoRedundantBackup": {
            "value": "Enabled"
        },
        "availabilityZone": {
```

```
            "value": ""
        },
        "vmName": {
            "value": "Standard_D16ads_v5"
        },
        "standbyAvailabilityZone": {
            "value": ""
        },
        "firewallRules": {
            "value": {
                "rules": [
                    {
                        "name": "AllowAllAzureServicesAndResourcesWithinAzureI
ps_2024-9-9_16-6-5",
                        "endIPAddress": "0.0.0.0",
                        "startIPAddress": "0.0.0.0"
                    }
                ]
            }
        },
        "network": {
            "value": {
                "publicNetworkAccess": "Enabled"
            }
        },
        "tags": {
            "value": {}
        },
        "version": {
            "value": "16"
        },
        "administratorLogin": {
            "value": "ptuser"
        },
        "administratorLoginPassword": {
            "value": null
        },
        "apiVersion": {
            "value": "2023-06-01-preview"
        },
        "storageAutogrow": {
            "value": "Enabled"
        },
        "iopsTier": {
            "value": "P30"
        }
    }
}
```

This project was commissioned by Microsoft and AMD.


Facts matter.®