

# POSTGRESQL @ AZURE

<https://vikasrajput.github.io>

PostgreSQL is an [OSS](#) RDBMS, an Ingres successor. Refer [Wikipedia](#) for its history. Per latest Global trends, it's ranked #2 for Dev Usage, #1 for Dev desiring it (Ref: Stack Overflow [Survey](#)).

This document (a) presents PostgreSQL/MSFT documentation pointers for Architects & other Tech roles (b) isn't warranted in any way and (c) not a replacement of vendor documentation.

## HOW TO USE THIS FIELD MAP?

If new to PostgreSQL, you can start with [Foundations](#), then check [References](#), and finally step through [WAF Pillars](#). Alternatively, feel free to jump through [Dev/Admin](#) or specific WAF [pillars](#).

## FOUNDATIONS

What's New? [Dev Build](#). [Stable Build](#). Or refer [Feature List](#)

PGSQL is one of the most stable, true OSS DBMS, and handled relational and large-scale data, including spatial and streaming. It gained lot of popularity since becoming a candidate for migration away from Oracle.

[PostgreSQL Internals](#), [Query Process: Connection, Parsing](#), [PgSQL Rule System](#), [Optimizer](#), [Executor](#)  
[DB Physical Storage: File Layout](#), [TOAST Storage](#), [DB Page](#)  
[Query Optimizer: Query Handling](#), [Algorithms](#), [GEQO](#). Index ([BTree](#), [GiST](#), [SP-GiST](#), [GIN](#), [BRIN](#), [Hash](#)), [Statistics](#)

## DEV

Not the absolute basic stuff when it comes to DB Dev.

[DDL \(RLS, partition\)](#), [DML](#), [Queries \(Windowing Function, CTE\)](#), [Functions](#), [Index](#), [Full Text Search](#), [Concurrency \(MVCC\)](#), [Performance Tips](#), [Parallel Query](#), [Large Objects](#), [Information Schema](#)

[Extend SQL, Rule System](#), PL: [PgSQL](#), [Python](#), [Tcl](#), [Perl](#)

## ADMIN

All these guides are very detailed. Refer or scan through, as preferred.  
[Config Params \(auth, consumption, log, vacuum, lock\)](#), [Client Authentication and Roles](#), [Manage & Maintain DBs](#), [Backup - HA/Replication - WAL](#), [Monitor \(DB, Disk, Test\)](#), [JIT Compile Background Worker Process](#), [Replication Progress](#)

[Additional Modules](#), [PostgreSQL Limits](#)

## POSTGRESQL TOOLS

Tools below are from MSFT or Oracle. 3rd Party products not included.  
[pgAdmin](#), [phpPgAdmin](#), [ora2pg](#), [VS Code - PostgreSQL](#)

## AZURE DATABASE FOR POSTGRESQL (PAAS)

While you can host a self-managed PostgreSQL VM (know its [merits and challenges](#)), MSFT has [Azure DB for PostgreSQL](#) PaaS offering – with [Single](#), [Flexible](#) and [Hyperscale](#) Server modes.

Check out deployment options for [Azure PostgreSQL IaaS vs PaaS](#), and [Single vs Flexible vs Hyperscale](#). Compared to Self-managed, there are [Limits on Single, Flexible and Hyperscale Server](#).

Compare [Single vs Flexible Server](#) and [Flexible vs Hyperscale](#).

Take a note of [Version Support Policy](#) and guidance on [Upgrading Single Server](#)

## REFERENCES

It's impossible to cover every bit of reference out there, let's focus on...

Skilling: [MSFT Learning Path](#), [Reference Architecture](#), [MSFT Community Blog](#), [Azure Latest Updates](#)

MSFT Offering: [Azure Database for PostgreSQL](#), [QuickStart Templates](#), [Azure CLI for PostgreSQL](#), Deploy Azure PostgreSQL using [ARM](#), [Bicep](#), [PS](#), [Terraform](#). Using [PostgreSQL for Enterprise Apps](#), [PostgreSQL Migration Guide](#), [Economic Value of Migrating PostgreSQL to Azure](#)

OSS/Vendor Resources: [PostgreSQL \(repo\) @ GitHub](#)  
[OSS @ Microsoft](#), [OSS Guidance for MSFT Teams](#)

## SECURITY

Security is the most critical WAF pillar. Let's get to it and categorize Security measures as Identity Management, Access Management, Surface Management, Protect Data and finally, Monitor/Audit.

	Single	Flexible	Hyperscale
Identity	<a href="#">AAD</a> , <a href="#">MFA</a> , <a href="#">Audit Logs</a>	<a href="#">Admin User</a> , <a href="#">SCRAM Roles</a>	<a href="#">Admin User</a>
Access			
Network	<a href="#">Private Link</a> , <a href="#">VNet Endpoint</a> , <a href="#">Firewall Rules</a>	<a href="#">VNet</a> , <a href="#">Private DNS</a> , <a href="#">IP Firewall Rules</a>	<a href="#">Private Link</a> , <a href="#">VNet</a> , <a href="#">Private DNS</a> , <a href="#">IP Firewall</a>
Protect	<a href="#">TLS</a> , <a href="#">CMK+AKV</a> , <a href="#">System Managed Disk Encryption</a>	<a href="#">TLS</a> , <a href="#">SCRAM</a> , <a href="#">System Managed Disk Encryption</a>	<a href="#">TLS</a> , <a href="#">System Managed Disk Encryption</a> , <a href="#">RLS</a>
Monitor	<a href="#">MSFT Defender</a> +		<a href="#">Azure Policy</a> , <a href="#">ATP</a> , <a href="#">Security Baseline for PostgreSQL</a>

## COST OPTIMIZATION

After Security, generally Cost holds the next priority for clients. Though it's very difficult to talk about this pillar in isolation. Firstly, every Org needs to establish a baseline Consumption (capacity, cost) [Forecast](#), [Budget](#) and [Ownership](#) and then be able to track and [Report](#) consumption. And then, we need to approach Architecture as such to elevate Demand Management (e.g., throttle) and Supply Management (e.g., scale).

[Azure Advisor](#), [Cost Management](#). Understand pricing for [Single](#), [Flexible](#) and [Hyperscale](#) Burstable SKU (flexible), [Reserve Capacity](#) (flexible, hyperscale)

## OPERATIONS EXCELLENCE

Ops Excellence proves the real-world agility and maturity of a business in managing Business Systems. At its core, Ops Excellence is all about how SDLC is managed, underscoring practices around Development, Deployment and Operation with Security, Monitoring and Automation embedded every step of the way.

Single: [PgBouncer](#), [Maintenance](#), [PostgreSQL Extensions](#), [Connection Libraries](#)

Flexible: [PgBouncer](#), [Maintenance](#), [PostgreSQL Extensions](#)

Hyperscale: [Server Groups](#), [Nodes](#), [PgBouncer](#), [Columnar tables](#), [Maintenance](#), [Extensions](#) (citus)

[Azure PostgreSQL GitHub Actions](#). [Azure PostgreSQL + AKS](#)

## RELIABILITY

Business should take lead on this and define must-have or preferred Availability and Recovery Metrics (SLA, MTTR, MTBF etc). This should inform Architecture – outlining High Availability (scale, prevent failure), Disaster Recovery (recognize failure, and recover) and Monitoring (service uptime, [chaos engineering](#), testing).

Single Server. HA: Protection against [Planned Outages](#) and [Unplanned Outages](#). DR: [Backup Recovery](#), [Read Replica](#), [Logical Decoding](#)

Flexible Server. HA: [Zone redundant](#) & [Same zone](#). Protection for [Planned/Unplanned Outage](#). Non-HA Server. DR: [Backup Recovery](#), [Restore](#) & [Networking](#), [Logical Decoding](#), [PgBouncer](#)

Hyperscale Server. HA: [Standby replica for each node](#). DR: [Backup Recovery](#), [Replica](#) (to read only server group) + [cross-region Replica](#), [PgBouncer](#)

Monitoring: [Monitor](#), [App Insights](#), [Service Health](#), [Resource Health](#), [Policy](#), [Advisor](#), [Alerts](#)

## PERFORMANCE

Performance is an interesting pillar – for it can be negatively impacted by almost all other pillars, but positive impact must be weaved in! Low-cost commitment or subpar operations or security measures can limit it. For this, establish Monitoring (workload, resources, baseline), Design for performance, and Remediate contention.

Monitoring. Most critical input in performance tuning is Baseline – workload & consumption. Workload expectation is business driven, and Consumption Baseline should be established (and adjusted) over a period.

[Monitor](#) (metrics, audit logs), [AppInsights](#), [QueryStore](#) (perf insights, perf recommendation), [Advisor](#), [Intelligent Tuning](#) (flexible)

Remediate: [Query](#), [CPU](#), [Memory](#), IO ([bulk inserts](#))