

# Best Practices

Essential guidelines for establishing secure and efficient connections to Vultr Managed PostgreSQL databases.

---

# Contents

01	Introduction	3
02	Why Use Connection Pools	3
03	Dedicated Pools Per Server	4
04	Connection Pool Modes	5
05	Pool Configuration Parameters	6
06	Conclusion	6

# Best Practices on Connecting to Vultr Managed Databases for PostgreSQL

## Introduction

Connecting to a Vultr Managed Databases for PostgreSQL becomes more efficient when you use connection pools. A pool maintains a set of reusable database connections, allowing applications to handle more requests without repeatedly creating and closing new connections. With proper configuration, connection pools improve performance, lower query response times, and ensure resources are used effectively.

This guide explains best practices for configuring and managing connection pools, including the importance of dedicated pools per server, the different pool modes, and key configuration parameters.

## Why Use Connection Pools

Every time an application opens a new PostgreSQL connection, the database performs setup tasks such as authentication and resource allocation. These operations introduce overhead and can significantly slow performance, especially under heavy application load.

A connection pool avoids this overhead by keeping a group of connections open and ready to use. When an application needs a connection, it borrows one from the pool and returns it when finished. This reuse allows queries to run faster, helps the system scale smoothly, and ensures resources are used efficiently.

- **Improved Performance:** Queries run faster because connections are already established, reducing the time spent on setup.

- **Resource Efficiency:** The database uses fewer resources since connections are reused instead of repeatedly opened and closed.
- **Scalability:** Applications can serve more users and handle higher traffic without overwhelming the database.
- **Simplified Management:** The pool automatically handles connection allocation and cleanup, reducing complexity in your application code.

## Dedicated Pools Per Server

Each server or application instance that connects to your Vultr PostgreSQL database should use its own dedicated connection pool. This approach provides:

- **Isolation:** Assigning a separate pool to each server prevents one server from consuming most of the available connections. This ensures that other servers and applications continue to run smoothly without unexpected slowdowns.
- **Predictability:** With individual pools, you can size each one based on the specific workload of its server. For example, a heavily used web server may require a larger pool than a background worker.
- **Scalability:** Dedicated pools make it easier to tune connection limits as you scale horizontally. Adding more application servers becomes straightforward because each one brings its own pool instead of competing for a shared pool.
- **Fair Resource Allocation:** When every server has its own pool, connections are distributed evenly. This avoids contention and ensures that all servers receive the database resources they need.

For example, if you run three application servers against the same database, configure a separate pool for each. This setup keeps the environment stable, predictable, and ready to grow.

# Connection Pool Modes

---

When you create a connection pool, you must choose a mode that controls how connections are allocated and reused. Each mode offers different trade-offs in terms of performance, predictability, and flexibility:

- **Session Mode (Recommended)**

- Each client gets a dedicated connection for the duration of its session.
- Best for most applications since it balances efficiency with predictable behavior.
- This mode works best for most use cases and is recommended if you are unsure which option to select.

- **Transaction Mode**

- A connection is assigned only for the duration of a transaction.
- After the transaction completes, the connection is returned to the pool.
- This mode fits workloads with many short-lived transactions that don't rely on session state.

- **Statement Mode**

- In this mode, the pool assigns a connection for a single SQL statement and immediately returns it after execution.
- This delivers the highest level of connection reuse but may break multi-statement transactions or features that depend on session variables.
- Use this mode only when handling simple, independent queries that don't require session context.

**Note**

If you are unsure which mode to use, go with **Session Mode**. It provides the most predictable behavior and works well for the majority of applications.

# Pool Configuration Parameters

---

When creating a connection pool in the Vultr Customer Portal or via the API, you must define several key parameters that control how the pool behaves:

- **Pool Name:** A unique identifier for the pool.
- **Database:** The logical database within your PostgreSQL instance that the pool connects to.
- **User:** The database user associated with the pool.
- **Pool Mode:** Determines how connections are allocated ( `session`, `transaction`, or `statement` ). Select the mode that best fits your application workload.
- **Size:** The maximum number of active connections in the pool. Tune this based on your application's concurrency requirements and server capacity.

For detailed, step-by-step instructions on creating a connection pool using the Customer Portal or the Vultr API, refer to the [Vultr PostgreSQL Connection Pools Documentation](#).

## Conclusion

---

You successfully learned how to optimize connections to your Vultr Managed Databases for PostgreSQL using connection pools. By reusing connections, you reduced overhead, improved query performance, and used resources more efficiently. Dedicated pools per server kept workloads isolated, predictable, and scalable. Selecting the right pool mode and tuning configuration parameters ensured consistent and reliable performance.



VULTR

