|  |
| --- |
|  |

**Model DB – Technical Document**

# Summary

This technical document provides a comprehensive breakdown of the PostgreSQL model database creation script, specifically tailored for AWS RDS environments. The model database (model) is intended as a reusable foundation for initializing new databases. It includes preconfigured schemas and structured permissions to ensure consistent, secure deployments across environments. Using this model-based approach eliminates manual setup errors, accelerates onboarding, and enforces a standardized schema and access structure for future databases.

# Technical Breakdown

1. **Database Existence Check**  
   The script first checks whether the model database already exists by querying the pg\_database system catalog. If it does not exist, the script creates it using the CREATE DATABASE model.
2. **Schema Initialization**  
   Upon creation, two schemas are established:

* public: PostgreSQL’s default schema.
* dba: a customer-facing administrative schema. It is used to surface relevant metadata, such as a customer’s own permissions or audit data. This enables secure transparency for clients while keeping core system logic separated..

1. **Permission Handling**  
   Permissions are explicitly controlled:

* All default privileges on the public schema are revoked to prevent insecure defaults.
* Ownership of both schemas is reassigned to secure admin roles.
* Role assignments include db\_datareader, db\_datawriter, db\_ddladmin, and db\_ddladmin\_c.
  + db\_ddladmin\_c mimics the core DDL admin role but is scoped to customer workloads to avoid cross-environment privilege spillage.

1. **Schema Creation Safety**  
   All schema creation logic is wrapped in dynamic DO $$ blocks to prevent errors if the schema already exists. This ensures idempotent behavior for repeatable deployments.
2. **Automation Compatibility**  
   The model database structure integrates with PowerShell and Git Bash workflows. These CLI tools can dynamically create customer-specific schemas and apply permissions programmatically without requiring GUI intervention.
3. **Reusability and Inheritance**  
   Any database cloned from the model db inherits its schema structure, default privileges, and access patterns. This removes configuration drift, simplifies CI/CD, and allows safe scaling across environments without sacrificing consistency.
4. **Customer Schema Integration (PowerShell/Git Bash Automation)**  
   The current deployment tooling—PowerShell and Git Bash—is used to dynamically create customer-specific schemas. These scripts automatically apply the appropriate permissions tied to the customer's scope and role assignment.

# SOURCES

* 1. <https://www.postgresql.org/docs/current/manage-ag-templatedbs.html>
  2. <https://www.ibm.com/think/tutorials/postgresql-tips-template-databases>

# TL;DR

The PostgreSQL model database is created with two predefined schemas (`public`, `dba`), custom permissions, and sets the DB as a template. It's designed for cloning new DBs quickly and consistently.