

Java/Oracle Application Migration Guide to Azure Database for PostgreSQL

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

© 2020 Microsoft. All rights reserved. This document is for informational purposes only. Microsoft makes no warranties, express or implied, with respect to the information presented here

Contents

[About the guide 6](#_Toc36107839)

[Scenario overview 6](#_Toc36107840)

[PostgreSQL Introduction 8](#_Toc36107841)

[Legacy on-premises application architecture 8](#_Toc36107842)

[Oracle Database ER Diagram 10](#_Toc36107843)

[Database schema objects to be migrated 11](#_Toc36107844)

[Get the document artifacts from Git repo 11](#_Toc36107845)

[How to install the Oracle database 11](#_Toc36107846)

[How to install the application 13](#_Toc36107847)

[Installing and running the Angular application 14](#_Toc36107848)

[Tour of the application 15](#_Toc36107849)

[Migration process 17](#_Toc36107850)

[Migrations types 17](#_Toc36107851)

[PostgreSQL database price considerations 19](#_Toc36107852)

[Database migration tool options 19](#_Toc36107853)

[Create your Azure resources 20](#_Toc36107854)

[Run the ARM template 20](#_Toc36107855)

[Update your secrets in Key Vault 21](#_Toc36107856)

[Update your web application configuration settings 21](#_Toc36107857)

[Setting up your migration server 23](#_Toc36107858)

[Choosing your migration server 23](#_Toc36107859)

[Hardware resources 24](#_Toc36107860)

[Securing the data during migration 24](#_Toc36107861)

[Download and install the Oracle database client library 25](#_Toc36107862)

[Set up the environment variables 25](#_Toc36107863)

[Set up the pgAdmin PostgreSQL database client 25](#_Toc36107864)

[Download and install the ora2pg utility 26](#_Toc36107865)

[Discovering and assessing the source database with ora2pg 27](#_Toc36107866)

[Prepping your database for export 27](#_Toc36107867)

[Create your ora2pg conf structure 29](#_Toc36107868)

[Add your Oracle and PostgreSQL DSN configuration to the conf file 30](#_Toc36107869)

[Testing your database connections and permissions 32](#_Toc36107870)

[Separating the constraints and indexes into files 33](#_Toc36107871)

[Evaluate the data type conversions 35](#_Toc36107872)

[PL/SQL to PL/pgSQL Syntax Conversion Guide 35](#_Toc36107873)

[Other useful ora2pg configurations 37](#_Toc36107874)

[Assessing database complexity and time to import 37](#_Toc36107875)

[Comparing the Oracle and PostgreSQL instance schema 39](#_Toc36107876)

[Migration 40](#_Toc36107877)

[Copying the data over to Azure PostgreSQL 40](#_Toc36107878)

[Post-migration 43](#_Toc36107879)

[Have questions? 43](#_Toc36107880)

## About the guide

Azure Database for PostgreSQL is a managed service that you use to run, manage, and scale highly available PostgreSQL databases in the cloud. This guide walks through the steps of migrating an on-premises legacy open source application using an Oracle database to using a cloud hosted Azure PostgreSQL database. Every application is different and has its own level of complexity. Very complex applications and databases require extra time and resources to analyze, assess the level of effort, and implement a solution. This guide focuses on simple to moderately complex web applications that need to switch the database to PostgreSQL. The application was architected to be simple, but realistic. Building a full application would take away from the process of understanding a database migration. The reader should appreciate the few changes required to retool the application in order to take advantage of the Azure Database for PostgreSQL . The Oracle database fields, objects, and data chosen represent a majority of the objects found in community projects. More complex Oracle types would put this application into the advanced migration scenario. Some of the database schema objects are contrived and meant to exercise the process of migration, not advocate best application architecture guidelines. The reader should have some basic familiarity with the development tools referenced and their usage. Step by step debugging instructions will not be provided.

## Scenario overview

Contoso Tech specializes in providing leading edge training and technical evangelism. They host multiple conferences around the world containing packed audiences. The website used for promoting the conferences, marketing session tracks, and registering attendees has been in production for several years.

Contoso has been modernizing their infrastructure and the operations team would like to move this application from on-premises to the cloud as it is one of the last applications in their shrinking datacenter. The operations team recently had to move the application and database to newer hardware. During the migration, it was discovered the database configuration was hard-coded in the application configuration. The application failed to start after migration and developers were called in to resolve. Since then, the development team has made great improvements to the web and API stack preparing for the move to the cloud. However, there is a reluctance to modernize or change the database feeding the web site due to the amount objects layered on over the years. A lot has changed in the database marketplace since this application was originally developed and there might be new database options to consider.

Management has heard about the benefits of the PostgreSQL database:

* High performance and scalability.
* Popular programming languages have a PostgreSQL provider.
* It can store many of the data types required by applications.
* It has an open-source licensing model.

Steps required to migrate to this database platform need to be explored by the development team.

The development team is motivated, but has communicated firmly migration is not as easy as changing a connection string in the application configuration. PostgreSQL has very similar capabilities to the existing Oracle database, but special considerations would be required for database objects that did not convert transparently. Downtime, functionality degradation, and loss of data are biggest risks for this project. The development team has been given the task of documenting the migration and risk mitigation plan.

The web application consists of an Angular frontend, Java Spring Maven web API, and an Oracle database.

This guide will cover the migration path for hosting an existing Java Spring Oracle application in Azure utilizing App Services and managed PostgreSQL PaaS service. The topics of application scaling and containerization are important, but will not be covered as the focus of this guide is the challenges of database migration.

## PostgreSQL Introduction

**Additional resources**

[Oracle to Postgres Conversion](https://wiki.postgresql.org/wiki/Oracle_to_Postgres_Conversion)

## Legacy on-premises application architecture

Since the legacy application is hosted on-premises, the developer could run this locally.



This sample application utilizes the following frameworks and components.

1. Angular 9 front end
2. Java SE 11 JDK
3. Maven 3.6.3
4. Sprint Boot 2.2.5 RELEASE
   1. Hibernate ORM
5. Embedded Tomcat
6. Oracle 11g
7. NodeJS
8. NPM

Microsoft supports multiple JDK versions. Check for your supported version.

**Additional resources**

[Java long-term support for Azure and Azure Stack](https://docs.microsoft.com/en-us/java/azure/jdk/?view=azure-java-stable)

[Azul Product Support Lifecycle](https://www.azul.com/products/azul-support-roadmap/)

## Oracle Database ER Diagram



## Database schema objects to be migrated

1. Primary and foreign field constraints.

2. Stored Procedures

3. Views

5. Triggers

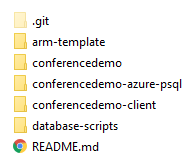
6. Indexes

7. Sequences

8. Field Data Types exercised: NUMBER, DECIMAL, VARCHAR2, DATE, CLOB, BLOB

## Get the document artifacts from Git repo

* Run git clone to download the application locally. You should have the structure below.



We are going to focus on setting up the Angular and Java application with a Oracle backend to provide the reader an understanding of the legacy application before the PostgreSQL migration tasks.

* **arm-template** template file to set up the entire Azure migration environment.
* **conferencedemo** folder contains Java API application.
* **conferencedemo-client** contains the Angular application
* **conferencedemo-azure-psql** contains the same Java API application with minor changes to connect to PostgreSQL.
* **database-scripts** contains the Oracle scripts to set up the database objects and sample data.

Once they have a contextual understanding of application, we will move on to the process of migration and conversion.

\*\* If you are utilizing a server other than a local copy, this account will need elevated permissions to this database in order to properly capture schema and data information for PostgreSQL export. If you are using a local Oracle copy, grant all the rights.



It will help demonstrate your typical challenges with assessing the entire database.

## Tour of the application

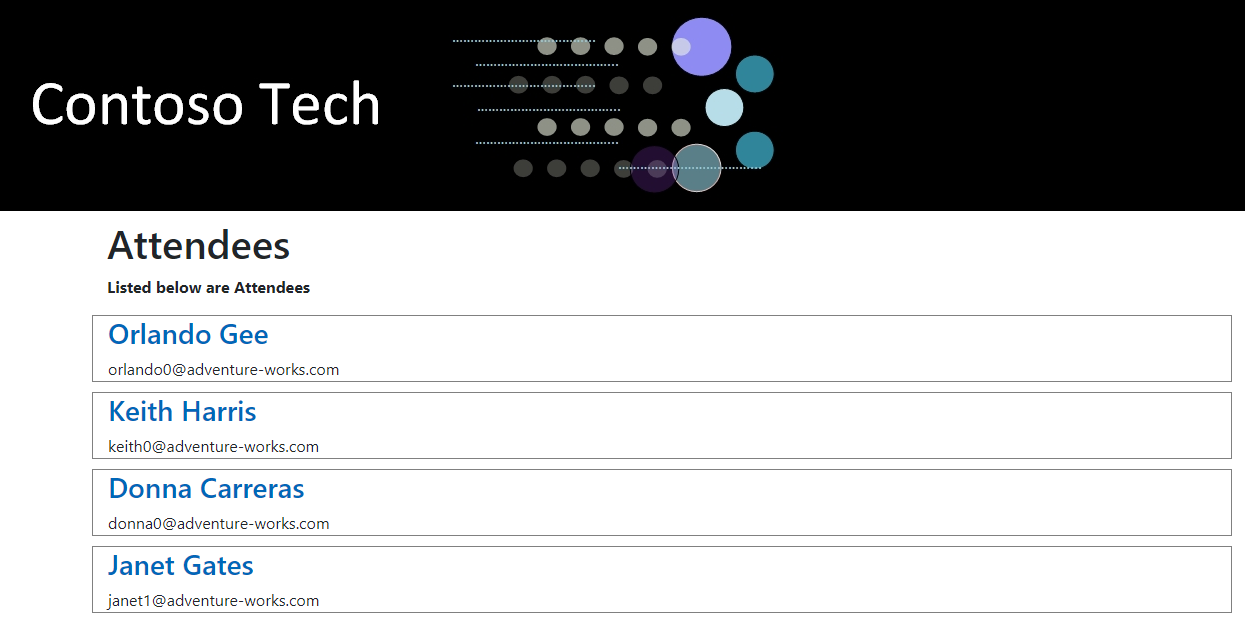
Landing page



After selecting an event, the session list presented. Attendees can register for the sessions on this page.



Administrators can view the attendees.



Speaker bio details



## Migration process

At this point, you should have the sample application running and be able to identify with the familiar architecture. We will be reference the parts of the project throughout the project.

The entire migration process can be broken down into these discrete phases.



## Migrations types

Each application migration needs to be evaluated and estimated based on its own merits. The types of projects fall into these categories generally.

* Little to no code changes. Migration works without issues.
* Some effort and code changes. Some schema objects require review and adjustments.
* Difficult and time consuming.

The process of moving an application to Azure should follow this maturity process.



**Additional resources**

[Application Modernization on Azure](https://medius.studios.ms/Embed/Video/BRK2102?sid=BRK2102)

## PostgreSQL database price considerations

* Scalability and pricing
* Single server vs Hyperscale
* Price calculator

<https://azure.microsoft.com/en-us/pricing/details/postgresql/server/>

When taking into account the size of a PostgreSQL database server, you need to account for the following items: performance, scalability, maintainability. Microsoft Azure Database for PostgreSQL has the capability to scale to your needs.

<TODO: Finish>

## Database migration tool options

<TODO: Finish>

* Azure Database Migration Services
* ora2pg utility
* Other commercial data transfer utilities
* Focus on ora2pg utility

## Setting up your migration server

This next section provides information related to setting up a server for database migration and the choices to consider.

### Choosing your migration server

#### Run locally or use a migration server?

You could run the ora2pg migration utility on your local development machine or the database server. The Oracle administrator will not appreciate the extra software installed on the server as well as the resources consumed during migration. Also, you would need to repeat this installation and configuration for each server. This approach is not recommended. Usually, a migration effort requires multiple team members. Running the migration locally on your machine would require you to create a setup document for the rest of the team in order for you to run the process in a similar fashion. This might not be efficient as project configuration and processing would need to be kept in sync. Also, running locally will consume significant resources causing the hardware to be tied to the migration process until completion. An alternative to running locally, would be to use a migration server(s). Multiple team members can check a central standardized migration server for progress and exception handling. It can be secured using best practices.



### Hardware resources

Your migration server should be reasonably configured with enough processing power and memory to handle the load. Memory is the key issue with migrating large amounts of data, especially records containing blobs. You may have reduce your data limit (rows processed per batch) significantly if you do not provide enough migration server resources. Receiving an out of memory error could cause unwanted project delays. The cost of delays may exceed the cost of moving to proper Azure SKU.

The migration server needs access to the Oracle and the Azure PostgreSQL instances. Depending on your project timelines, you may need to increase your throughput between the source environment and the Azure PostgreSQL network.

### Securing the data during migration

On-premises migration server

Encrypting your data during migration is critical. This can be done utilizing a few methods:

* Database provider connection <TODO: Investigate details>
* VPN gateway
* ExpressRoute

Make sure to factor in the calculation of encryption into the total transfer time. Your local spike testing results may differ from actual production data processing.

Azure hosted VM

After provisioning the VM and Azure Database for PostgreSQL, two configurations are needed for enabling connectivity between them: “Allow Azure Services” and “Enforce SSL Connection”, depicted as follows:

* “Connection Security” blade -> Allow access to Azure Services -> ON
* “Connection Security” blade -> SSL Settings -> Enforce SSL Connection -> DISABLED

### Download and install the Oracle database client library

If you are running the Oracle XE database locally, then you can you skip the install of the database client.

### Set up the environment variables

#### Windows server

ORACLE\_HOME = <Your client or server install path>

LD\_LIBRARY\_PATH = %ORACLE%\lib

\*\* If you installed the database client only, then set your LD\_LIBRARY\_PATH path to <client install path>.

Example



### Set up the pgAdmin PostgreSQL database client

You will need to install the pgAdmin admin client. You can download the utility from <https://www.pgadmin.org/download/> . Connect to the Azure database server with the server information captured earlier.

Create a database **conferencedemo** database.



Create a user **reg\_app**. Assign the **conferenceadmin** role.



### Download and install the ora2pg utility

**<TODO>**

**Additional resources**

[Migrate Oracle to Azure Database for PostgreSQL](https://datamigration.microsoft.com/scenario/oracle-to-azurepostgresql?step=1)

[What is VPN Gateway?](https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpngateways)

[Virtual machine network bandwidth](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-machine-network-throughput)

[Optimize network throughput for Azure virtual machines](https://docs.microsoft.com/bs-cyrl-ba/azure/virtual-network/virtual-network-optimize-network-bandwidth)

[ExpressRoute overview](https://docs.microsoft.com/en-au/azure/expressroute/expressroute-introduction)

## Discovering and assessing the source database with ora2pg



### Prepping your database for export

Before running the ora2pg utility against your source database, the database statistics will need to be updated. Statistics can become stale over time because of changing data volumes or changes in column values. Statistics can be inaccurate after lots of data and schema changes.



Before running the ora2pg utility against the source database, you need to check for invalid objects. The Datamigration Team at Microsoft wrote an Oracle procedure that queries the database and shows the count of objects and their validity. Invalid objects will not be converted by the ora2pg utility by default.



### Create your ora2pg conf structure

For small projects, running the ora2pg utility with the defaults will allow you to export all objects in one giant script. For larger more realistic projects, you will be running the ora2pg utility several times. It is important to separate your scripts into easily maintainable directories. You will want to import database objects and data into PostgreSQL in stages. It will be a rare project where you can accept all the defaults and run the data migration process.

Using Git, the team can track changes to the schema and make development decisions based on those changes. Separating your scripts into directories makes it easier to review the changes.

To create your base project directory structure using the ora2pg tool:

1. Navigate to a base directory.
2. Run this command

ora2pg --init\_project reg\_app



Now that your base structure has been created, place your **ora2pg conf** file into the config folder.



### Add your Oracle and PostgreSQL DSN configuration to the conf file

Configure Oracle: ORACLE\_HOME, ORACLE\_DSN, ORACLE\_USER, and ORACLE\_PWD information.

Example of DSN settings



If you are having trouble finding your DSN settings, try running ‘lsnrctl status’ command in the console.

You can confirm:

1. ORACLE\_HOME value
2. Oracle SID
3. Oracle host and port



Configure the Azure Database for PostgreSQL:

PG\_DSN, PG\_USER, PG\_PWD



### Testing your database connections and permissions

#### Set your schema

Run the following command **if you created a local Oracle XE test server**. If you are using a database on a server with several other databases, skip ahead.



434 tables?! What happened? Why are the apex tables in the output?



You need to specify the schema/namespace.



If you know are going to work with one schema, you can set it in the conf file.



This is the output we were expecting, but we still have a problem. All of the constraints and indexes are in the table creation scripts.



If you have a lot of data, the indexes and constraints should be applied once the tables have been created and the data has imported. You will have much better data import performance.

### Separating the constraints and indexes into files

Update your conf file with the following configurations.





This type of schema output allows the database tables to be created first. Data could be imported and then the indexes and constraints can be applied at a later time.



All the tables schemas can be found in the ‘reg\_app-psql.sql’ file. Notice the indexes and constraints are missing.



A database with many tables or tables with many fields may require a different migration strategy. You could create a script for each table. This will allow team members, like the database administrator, to evaluate the conversions before migration.





### Evaluate the data type conversions

Most of the ora2pg type conversions suggestions make sense. There are times the migration team will need to adjust the schema data types based on knowledge. Evaluating each table and its dependent objects makes sense. Do I need a bigint or is int good enough?



### PL/SQL to PL/pgSQL Syntax Conversion Guide

<TODO: Explain high level problems>

Examples

Oracle evaluation of ‘’ and NULL.

<TODO: Point to other references.>

**Additional resources**

<https://www.postgresql.org/docs/11/sql-syntax.html>

[Oracle to PostgreSQL Wiki](https://wiki.postgresql.org/wiki/Oracle_to_Postgres_Conversion)

### Other useful ora2pg configurations

Your ora2pg conf file has other useful configurations. They are documented in the conf file. Read the descriptions carefully as they have specific valuable warnings. This may save you time as the migration process may error out on large tables if the configurations are not correct.

|  |  |
| --- | --- |
| Parameter | Value |
| LOG\_ON\_ERROR | 1 |
| FILE\_PER\_CONSTRAINT | 1 |
| FILE\_PER\_INDEX | 1 |
| FILE\_PER\_FKEYS | 1 |
| FILE\_PER\_TABLE | 1 |
| DATA\_LIMIT | 2000 |
| FORCE\_OWNER | 1 |
| BLOB\_LIMIT | 500 |
| TRUNCATE\_TABLE | 1 |

### Assessing database complexity and time to import

It is important to understand how the complexity involved with migrating your database. The ora2pg utility has a built-in report that can assist with this task.



Example





This application is considered to be an easy migration.

### Comparing the Oracle and PostgreSQL instance schema

After you create the tables in the PostgreSQL source database, it may be time to verify all of the objects were created as expected. Ora2pg provides a useful feature to easily compare the two databases.

ora2pg -c config/ora2pg\_dist-my-migration-test.conf -t TEST --namespace REG\_APP>migration\_diff.txt

Example



## Migration

Once your team has an opportunity to review the converted schema and decide if the default choices were acceptable, it may be time test the migration.



### Copying the data over to Azure PostgreSQL

In this example, only the **SESSIONS** table is being transferred to PostgreSQL. This type functionality is helpful if you are required to carefully verify each table. Tables with millions of rows may need this type process.

When migrating data, always COPY data export mode. You get a performance boost compared to using the INSERT command.



Example of the PostgreSQL table updated



The default ora2pg settings for COPY could be used against a simple database that does not have much data.



Notice the data was copied over to the PostgreSQL database. This included the blob and clob fields.



Tables with simple number columns can copy millions of rows quickly. Tables containing blobs will take much longer, especially if the blobs are large.

Running the script multiple times can cause duplicate data to be inserted into the target database.



## Post-migration

****

**Additional resources**

[Oracle Inventory Script Artifacts](https://github.com/microsoft/DataMigrationTeam/tree/master/IP%20and%20Scripts/Oracle%20Inventory%20Script%20Artifacts/Oracle%20Inventory%20Script%20Artifacts)

## Have questions?

For any questions or suggestions about working with Azure Database for PostgreSQL, send an email to the Azure Database for PostgreSQL Team ([@Ask Azure DB for PostgreSQL](mailto:AskAzureDBforPostgreSQL@service.microsoft.com)). This address is for general questions rather than support tickets.

In addition, consider these points of contact as appropriate:

* To contact Azure Support or fix an issue with your account, [file a ticket from the Azure portal](https://portal.azure.com/?#blade/Microsoft_Azure_Support/HelpAndSupportBlade).
* To provide feedback or to request new features, create an entry via [UserVoice](https://feedback.azure.com/forums/597976-azure-database-for-postgresql).