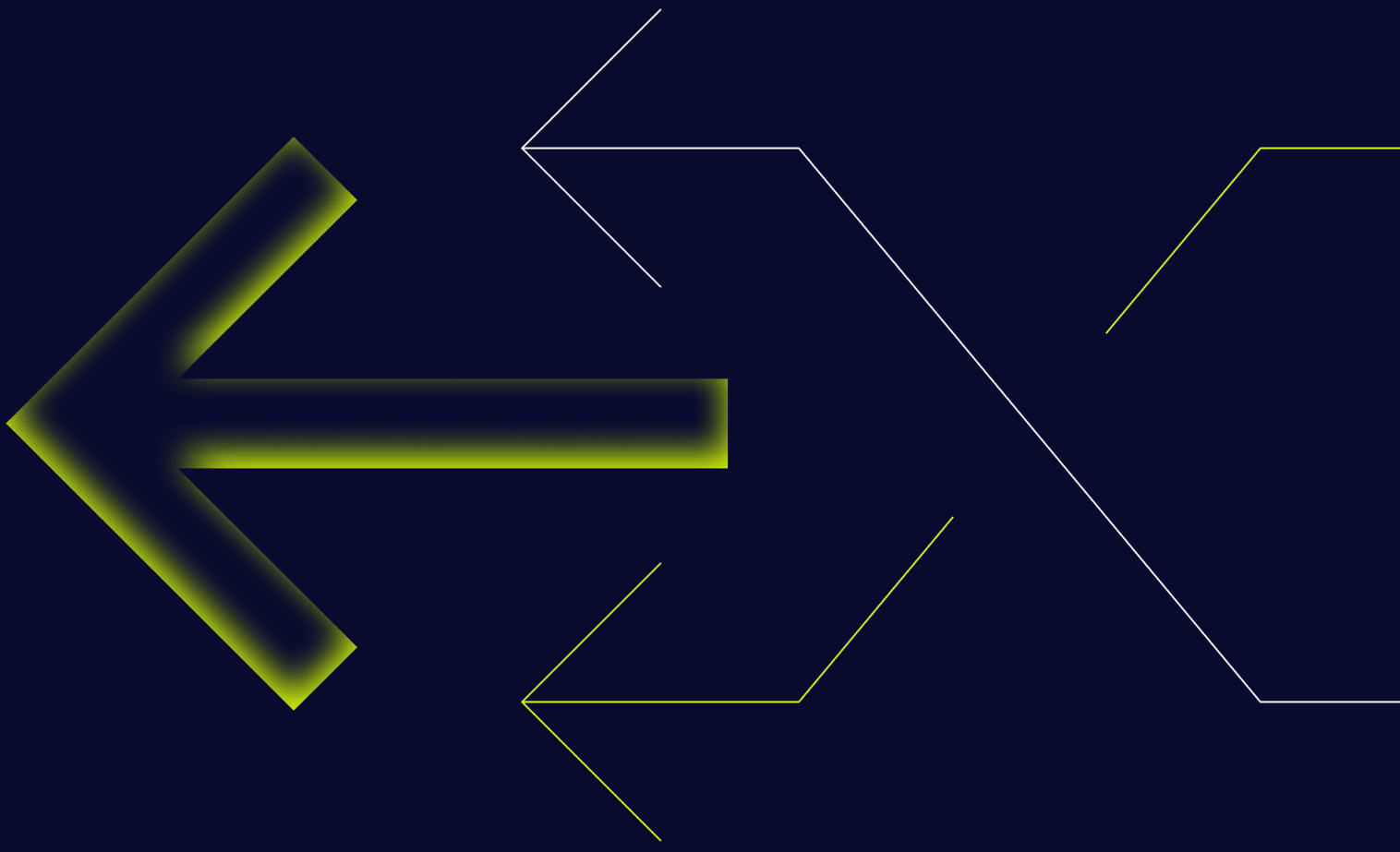
sonatype

Sonatype Security Research

OrientDB to PostgreSQL Migration

Reference Guide



Support for OrientDB databases with Nexus Repository is currently in **extended maintenance**, and will be officially sunset at the start of 2026. At that time, Sonatype will no longer support deployments using OrientDB. Database migrations often require some planning, so the time to begin preparing is now.

This document guides users through the process of migrating from OrientDB to PostgreSQL. The information here is pulled from our documentation, our Support Knowledge Base, and our Migration Specialists.

Jump Ahead

- **Readiness Flowchart:** Determine at-a-glance if you're ready to begin migration.
- **Migration Procedures:** Follow our migration procedures and make the jump from OrientDB to PostgreSQL.
- **Tips & Tricks:** Solve potential problems preemptively.
- **Troubleshooting:** Identify common problems and error messages, and their solutions.

Why PostgreSQL?

Although it's possible to migrate to an H2 database, **Sonatype strongly suggests that customers move to PostgreSQL.** PostgreSQL has a number of important advantages.

- PostgreSQL supports hot backups, high availability architectures, and cloud-native scaling, making it ideal for production and enterprise environments.
- Performance with PostgreSQL is generally better than OrientDB or H2, with faster queries, reduced latency, and better resource efficiency.
- Key features of Nexus Repository Pro — including some of the most powerful enterprise capabilities — require PostgreSQL. Future enhancements are being built exclusively for PostgreSQL users. Staying on OrientDB or moving to H2 means you'll miss out on innovation.

Reach Out to our Migration Specialists

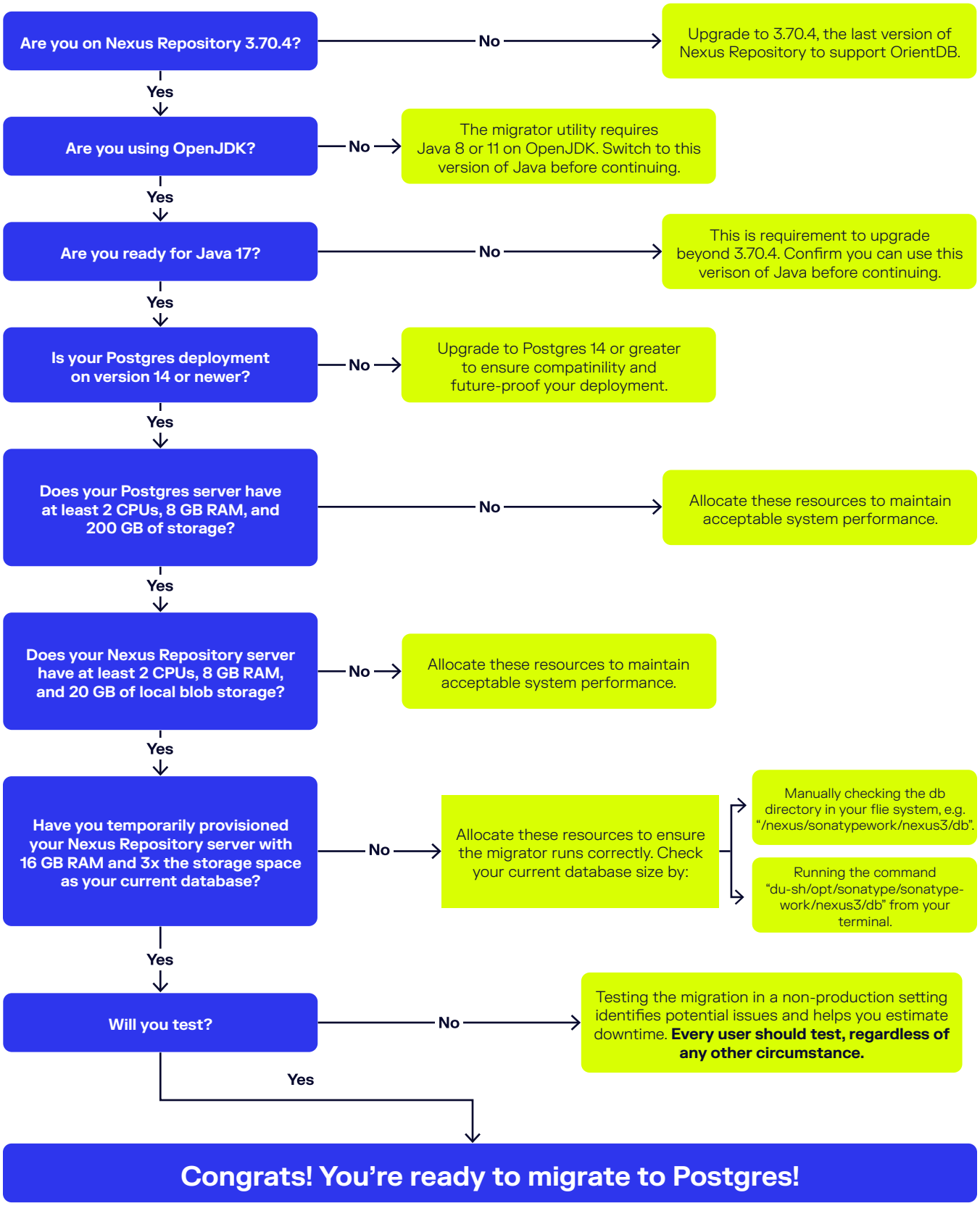
Users who have purchased Nexus Repository Pro, are on OrientDB, and are ready to migrate may have the opportunity to work directly with our Migration Specialists for hands-on guidance. If you have a Customer Success representative assigned, you'll receive an email with more information. If you don't, reach out to your account representative.

Repo Cloud

Migration is an opportunity to rethink the specifics of your deployment and consider the future. Nexus Repository Cloud is a SaaS deployment of Nexus Repository – ***we*** build the architecture, stay on top of versioning, and manage the database, and ***you*** reap the benefits. Reach out to your Sales or Customer Success representative to learn more.

Readiness Flowchart

Migration requires your deployment and environment to meet certain criteria. Review the flowchart below and be sure you can answer "yes" to all of the questions below before continuing.



Migration Procedures

If you meet all the migration prerequisites, then follow these steps to migrate.

Remember, to isolate problems and estimate the actual time requirements, **test** this procedure with a backup before attempting it in production.

- [ ] Perform a full backup of your Nexus Repository installation, including all directories and files.

  This backup will serve as a restore point in case of upgrade failure, but also protects you against unintended file system changes. See "Identify important directories before beginning."

- [ ] Upgrade to Nexus Repository 3.70.4 following normal upgrade procedures. Check our documentation.

  This version of Nexus Repository is the last to support OrientDB and Java 8 & 11, which OrientDB requires. Upgrading to this version ensures the smoothest possible migration.

- [ ] Download our migrator utility.

  Visit our downloads page for the migrator utility for Nexus Repository version 3.70.4. The exact version of the migrator utility, linked here, is required for this migration. Other versions of the utility will fail.

- [ ] Create a user and database in your PostgreSQL deployment.

  Create a user in your PostgreSQL deployment named nexus.

  ```
  CREATE USER nexus WITH PASSWORD '<nexus_password>' CREATEDB;
  ```

  Then create a database named 'nexus', with your new user as the owner.

  ```
  CREATE DATABASE nexus OWNER nexus ENCODING 'UTF8';
  ```

  Then give the user privileges to use the public schema and the database.

  ```
  GRANT ALL PRIVILEGES ON SCHEMA public TO nexus;
  ```

  ```
  GRANT ALL PRIVILEGES ON DATABASE nexus TO nexus;
  ```

- [ ] Configure PostgreSQL autovacuum configuration to be on.

  Adjust this setting in your `postgresql.conf` file. Remember that this setting also requires `track_counts` to be on, which you'll also adjust in that file.

  ```
  autovacuum = on
  ```

- [ ] In your `$data-dir/etc/fabric` directory, create a file named `nexus-store.properties` and populate it with the following:

  ```
  username=nexus
  password=<nexus_password>
  jdbcUrl=jdbc\:postgresql\://<database-host>\:<database-port>/nexus
  ```

Migration Procedures (Continued)

- [ ] For large databases, increase the maximum number of connections from 100 to 200.

  Find this setting in your `postgresql.conf file.`

  ```
  max_connections = 200
  ```

  Also increase the connection pool size from 100 to 200 by adding the following to `nexusstore.properties.`

  ```
  advanced=maximumPoolSize\=200
  ```

- [ ] Add the following to `$data-dir/etc/nexus.properties`

  ```
  nexus.datastore.enabled=true
  ```

- [ ] Perform a full backup using Nexus Repository's backup task.

  This backup is for the migration and does not serve as a restore point in case of failure.

- [ ] Move the backup to a clean working location, somewhere outside of Nexus Repository's `db` directory. Also move the migrator utility to this clean working location.

- [ ] Start Nexus Repository.

- [ ] Run the following command from the directory containing your database backup.

  ```
  java -Xmx16G -Xms16G -XX:+UseG1GC \

  -XX:MaxDirectMemorySize=28672M \

  -jar nexus-db-migrator-*.jar \

  --migration_type=postgres \

  --db_url="jdbc:postgresql://<database URL>:<port>/
  nexus?user=nexus&password=<nexus_password>"
  ```

  If you're using Java 11, you'll need this additional parameter.

  ```
  --add-exportsjava.base/sun.nio.ch=ALL-UNNAMED
  ```

- [ ] Run the following command on the new database's server.

  ```
  VACUUM(FULL,ANALYZE,VERBOSE);
  ```

- [ ] Start Nexus Repository.

Complete the Migration

Your migration isn't finalized just yet! Check the following four items before moving on.

- [ ] Allow Nexus Repository to complete the `Rebuild repository browse` and `Rebuild repository search` tasks. These tasks run automatically after migration. Do not shut down Nexus Repository until these tasks complete.

- [ ] If your Nexus Repository instance has a repository in the Helm format, manually run the task `Rebuild Helm metadata`.

- [ ] Check the task log and confirm all of the tasks below are complete.

    -repository.rebuild-index

    -repository.search.update

    -create.browse.nodes

    -repository.yum.rebuild.metadata

    -component.normalize.version

    -repository.metrics.blob.size.copy

    -file.blobstore.metrics.datastore.migration

- [ ] Install the trigram module to ensure Nexus Repository can accurately search components. On the PostgreSQL server, install, create, and then check the extension installed correctly with:

```
sudo dnf install postgresql-contrib
```

```
create extension pg_trgm
```

```
select * from pg_extension;
```

Tips and Tricks

Test with a backup before migrating your production instance
It is ***absolutely critical*** that you test the migration procedures with a backup before attempting to migrate in-prod. Testing is the only way to identify potential problems, estimate time/downtime requirements, and be certain that no data will be lost or compromised during the migration.

Identify important directories before beginning
Identifying key directories before starting migration will prevent confusion. Most deployments include two main directories:

- The installation directory, also called the application directory. You can identify this directory by the `NOTICE.txt and OSS-LICENSE.txt` files located here. In a typical installation, the location is `\nexus\nexus-3.70.4.`

Tips and Tricks (Continued)

- The data directory, also called the work directory. You can identify this directory by the subdirectorylabeled db, which is your OrientDB database. In a typical installation, the location is `\nexus\sonatype-work\nexus3.`
- Sometimes we indicate the data directory with `$data-dir, e.g. $data-dir/etc fabric`.

We recommend you start your migration procedure by backing up both the installation and work directory, including all subdirectories.

Plan for downtime
Migration requires a period of downtime, where Nexus Repository won't be reachable by users or systems. If possible, plan to migrate during off-hours. Announce the downtime well in advance.

The only way to accurately estimate the downtime is to test on a backup before attempting the migration in-prod.

Migration is one-way
Migration is **one-way.** You cannot migrate from OrientDB to PostgreSQL and then migrate back to OrientDB.

However, migration is also **nondestructive.** Your old OrientDB database remains in your data directory and can be used as a restore point, if that becomes necessary.

Ensure ownership is correct

If you're using a Unix-based system, then the migration procedure may have changed the OS-level owner of certain files, which may prevent a smooth startup or create runtime permission issues.

To prevent this, **ensure all files are owned by the service user under which the service is running**. Set permissions recursively with the `chown` command. That might look like:

```
sudo chown -R nexus:nexus /opt/nexus
```

Recognize unsupported formats
Nexus Repository + PostgreSQL does not support Bower or any of the community-created formats. Data in these formats will not be migrated by this procedure.

Review configuration options
Two key PostgreSQL configuration options can have a big impact on the specifics of your operation. Review both these configuration options and make changes to your PostgreSQL database, as necessary.

- For large deployments, adjust the maximum pool size to 200. This improves performance even under heavy loads.
- If you're using orchestration tools, relational database services, or other infrastructure to launchNexus Repository, adjust the max lifetime to be several seconds shorter than any infrastructureimposed connection time limit. This prevents connection errors and maxmizes the benefits of orchestration tools.

Migrate in the same datacenter
If downtime during the migration is a concern, perform the migration while the Nexus Repository server and the PostgreSQL server are in the same datacenter or availability region. This maximizes the performance of the migration tool, safeguards against interruptions, and ensure the smoothest possible migration.

Upgrade, then Migrate, then Upgrade
It's not possible to migrate databases while simultaneously upgrading your Nexus Repository instance. if you start migration on version 3.70.4, you must also end migration on version 3.70.4.
If upgrading to the latest version of Nexus Repository is important to you (and it should be!), then:

1. Upgrade Nexus Repository to 3.70.4.
2. Migrate from OrientDB to PostgreSQL.
3. Upgrade to Java 21.
4. Upgrade Nexus Repository to the latest.

We **strongly recommend** you make time to upgrade Nexus Repository regularly. This ensures you're receiving the value of new features and stability/performance improvements. Recognize that **Java 21 is required starting with 3.87.0**.

Review cleanup policies after migration
Postgres supports advanced search operations, and Nexus Repository takes advantage of this during cleanup tasks. After migrating, double check that cleanup policies are accurately identifying the correct components.

Prepare for new backup procedures
Backup procedures for embedded databases like OrientDB and external databases like PostgreSQL differ in a few key ways. Review our documentation on backing up and PostgreSQL's documentation for backing up before the migration and ensure you understand the new backup procedures.

Remember that Nexus Repository backups require the `blobs` directory and the keystores/node directory. Migrating to PostgreSQL does not change the location of these directories; by default, they're located in your data directory, e.g. `nexus\sonatype-work\nexus3`.

Consider resilient architecture
One of the major benefits of PostgreSQL as a database option is that it's compatible with resilient and high-availability architectures, which are critical for large production deployments. Nexus Repository is a critical part of your software supply chain, and downtime – intentional or otherwise – hurts productivity, creates tech debt, and hurts value.

Once you're on PostgreSQL, begin considering resilient or high-availability architecture. Refer to our reference architectures and deployment pattern library to review our recommendations.

Troubleshooting

Repositories appear empty after migration
Run the `Repair - Rebuild repository browse` task to rectify potential discrepancies between the data model that handles browsing and the data model that handles component info. If the problem persists, try searching for a component directly and/or run a build against a repo to verify that components.

If both of these fail, it's possible that migration failed. If you're a customer, open a Support ticket. If not, restore a backup with the OrientDB database and try migration again.

Read more in our Support knowledge base.

Troubleshooting (Continued)

“Value too long for type character varying(250)”
The `db_url` value in your migration command is too long. A sample migration command is below, with the `db_url` value in red.

```
java -Xmx16G -Xms16G -XX:+UseG1GC \

-XX:MaxDirectMemorySize=28672M \

-jar nexus-db-migrator-*.jar \

--migration_type=postgres \

--db_url=”jdbc:postgresql://<database URL>:<port>/
nexus?user=nexus&password=<nexus_password>”
```

This error is likely if you’re specifying additional parameters to enable SSL features. Shorten the value to below 250 characters and try the command again.

Read more in our Support knowledge base.

“DB Files Detected”
Your Nexus Repository backup is located in your `db` directory. Move the backup to a clean working location and run the migration command again.

Nexus Repository can’t reach the Postgres server
Check that Nexus Repository can communicate with the Postgres server, and that its privileges are adequate, by running the command `psql` from the server hosting your Repository Nexus application.

“Permission denied for public schema”
The PostgreSQL user representing Nexus Repository (**nexus**, if you’re following this guide) does not have appropriate privileges. Ideally, the user nexus will be the owner of the database. Alternatively, give the user the **CREATE** and **USAGE** (or the equivalent **ALL**) privilege. The command looks like:

```
GRANT ALL PRIVILEGES ON SCHEMA public TO nexus;
```