PEGA PLATFORM

Pega Platform 8.1

Installation Guide

For Tomcat and PostgreSQL

Trademarks

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

Notices

This document is the property of:

Pegasystems
One Rogers Street
Cambridge, MA 02142-1209, USA
Phone: 617-374-9600 Fax: 617-374-9620

www.pega.com

Document: Pega Platform 8.1 Installation Guide

Feedback

If you have comments for how we can improve our materials, send an email to DocRequest@Pega.com.

Contents

Overview and system requirements

This guide describes how to install a new instance of Pega Platform. See the *Platform Support Guide* on the Pega Community for a list of supported platforms.

To upgrade, see the *Pega Platform Upgrade Guide*.

Installing Pega Platform is a multiple step process that involves configuring your database and application server, loading rules into the database, and then deploying application archives to the application server.

Pega Platform supports different deployment topologies and configuration options that affect how the supporting infrastructure is configured and managed after installation. Engage your database administrator and any other infrastructure resources as soon as possible in the planning process.

Pega Platform architecture

Pega Platform is a Java EE-compliant enterprise application that requires an application server and a database server:

- The application server hosts the Pega Platform application archives and provides interconnectivity to other systems through various protocols.
- The database server stores the rules, data, and work objects used and generated by Pega Platform.

Application users and developers typically access Pega Platform through a web browser. Applications can also expose HTTP-based services (for example, SOAP, REST, or HTTP) for administration or process automation in a headless environment.

Plan your deployment

Pega Platform supports several configuration options that can affect the choices that you make during the deployment. Before beginning, read this section thoroughly.

Plan your architecture and configuration.

- Choose a configuration type: single-schema or split-schema configuration. Pega recommends a split-schema configuration. See Split-schema and single-schema configurations.
- Choose whether to use the standard product edition or the multitenancy edition. The multitenancy edition has different requirements, different run-time behaviors, and different administrative procedures from the standard edition. Before you select the multitenancy edition, review the *Multitenancy Administration Guide* on the Pega Community.

  Upgrading from one edition to another is not supported. The schema DDLs for the two editions are not compatible. For example, if you install the standard edition and later decide to use the multitenant edition, you must either drop and re-create the database or create a new database.
- Choose whether to use Kerberos functionality. Kerberos is a computer network authentication protocol that allows nodes communicating over a non-secure network to prove their identity to one another in a secure manner. If you enable Kerberos authentication, you must use the command line method to deploy Pega Platform. For more information, see your installation guide.

Consider the following precautions before you continue:

- Always maintain a backup of your system, especially before performing an upgrade.
- Do not change your environment while you are deploying Pega Platform. For example, if you are making changes to your application server or database server, do so before you deploy Pega Platform.

- The upgrade process requires additional space approximately equal to twice the size of your rules schema. Ensure that your database can allocate enough space to perform this upgrade.

Split-schema and single-schema configurations

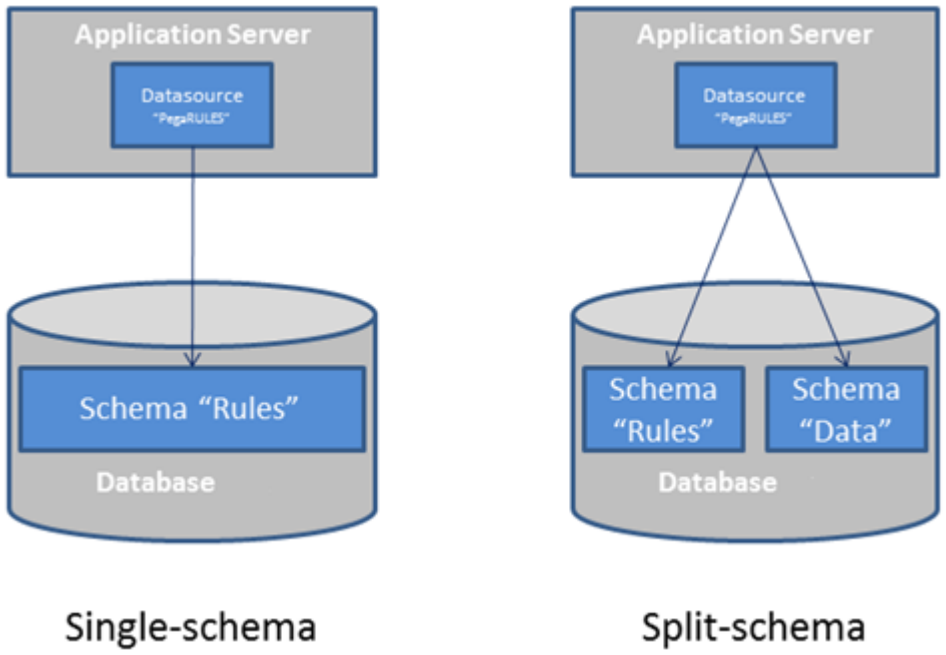
There are two configuration types: single schema and split-schema. Pega recommends split-schema configurations, particularly in critical development environments such as quality assurance, staging, and production.

- Single-schema configuration — One schema contains all rules and data objects.
- Split-schema configuration — The rules and data objects reside on separate schemas:
  - A Rules schema contains rules tables and associated data and objects.
  - A Data schema contains transaction data, including work objects.

  With a split-schema configuration, you can upgrade one environment, and then migrate the upgraded objects to other environments.

  In a split-schema configuration, Pega Platform uses the Java Naming and Directory Interface (JNDI) standard to identify and access the appropriate schema. One of the benefits of using JNDI is that it allows Pega Platform to access different schemas while using only a single data source.

  The following diagram illustrates the difference between a single-schema configuration and a split-schema configuration.



Single-schema Split-schema

If you plan to use a Pegasystems-supplied application and would like to store any non-Pega-specific data in an separate schema, you can optionally configure a separate customer data schema in addition to the default Pega data schema.

Deployment methods

You can deploy Pega Platform either with the UI tool or from the command line. This guide includes instructions for both methods.

- UI tool – Use the UI-based Installation and Upgrade Assistant to install either the rulebase or the rulebase and the schema.

- Command line – Run scripts to deploy Pega Platform.

Regardless of which method you use, you might need to edit the **setupDatabase.properties** file that controls the behavior of several scripts:

- The **generateddl.bat** or **generateddl.sh** script generates an SQL file that your database administrator can use to apply schema changes to the database. You can run this script regardless of whether you use the IUA or the command-line script.
- The **install.bat** or **install.sh** script performs the following functions:
  - Deploys the most recent version of Pega Platform.
  - Specifies whether to generate a DDL file of changes to the database.
  - Enables Kerberos authentication.

  If you use the IUA to install, you do not use the **install.bat** or **install.sh** script.

Review the documentation

Before you begin the installation, review the related information available on the Pega Community.

- Review the *Platform Support Guide* before you install Pega Platform to verify that your database and application servers are supported.
- Review the important information in the Release Notes on the Pega Community before you continue.
- Review the help system.
- If you plan to configure Pega Platform in multitenant mode after deploying, review the *Multitenancy Administration Guide*.
- For Business Intelligence Exchange (BIX), review the *Business Intelligence Exchange User Guide*. BIX is included in the full distribution image, but has a separate installer.

Transport-layer encryption method

Pega recommends that you use a strong transport-layer encryption method (for example, Transport Layer Security 1.2) to secure Pega Platform web applications. This encryption requires that you create and install transport-layer security and secure socket layer digital certificates on your application server for Pega Platform.

Before you continue, determine what transport-layer encryption method you will use. For more information, see the documentation for your application server.

Node classification in high availability systems

Optimize performance and provide higher scalability and stability in a cluster by using node classification, which is the process of separating nodes, segregating them by purpose, and predefining their behavior.

Note: Node classification applies to high availability cluster environments only.

By configuring a node with a node type, you dedicate the node to perform particular actions and run only those agents, listeners, job schedulers, and queue processors that are mapped to the node type. For example, if a set of nodes is dedicated to user requests, background processes can be disabled to improve performance.

Every node that is started with the same node type uses the same template and follows the same behavior.

For more information, see *Node classification* on the Pega Community.

System requirements

Before you deploy, ensure that your system meets the following minimum requirements.

UI-based tool requirements

If you plan to use the UI-based Installation and Upgrade Assistant, ensure that the system meets these minimum system requirements in addition to all other requirements.

- 1.25 GB minimum available memory
- 10 GB minimum disk space plus at least 8 GB available space in the temporary directory of the root file system. The default temporary directory for the deployment is java.io.tmpdir.
- Java Platform, Standard Edition Development Kit (JDK)

Application server requirements

Install only Pega Platform on the application server. The application server must meet the minimum requirements listed in the *Platform Support Guide* on the Pega Community and in this section.

- Supported 64-bit JDK. See the *Platform Support Guide* on the Pega Community for a list of supported versions.
- 1 GB minimum free disk space. You might need additional storage space for debugging and logging.
- Memory requirements: Pega Platform runs in memory (heap) on Java Virtual Machines (JVMs). In general, all activity is distributed over multiple JVMs (nodes) on the application server.
  - Standard suggested system heap size is 4 - 8 GB based on monitoring of memory usage and garbage collection frequency.
  - Larger heaps are advisable if your applications allow a high number of concurrent open tasks per session or cache a large collection of transaction or reference data.
  - Do not deploy Pega Platform in an environment where the heap size exceeds the vendor-specific effectiveness limit.
  - The host application server memory size must be at least 4 GB larger than the Pega Platform heap size to allow space for the operating system, monitoring tools, operating system network file buffering, and JVM memory size (-XMX option). The minimum host application server memory size is 8 GB:

    4 GB heap + 4 GB for native memory, operating system, and buffering

    If the server does not have enough memory allocated to run Pega Platform, the system can hang without an error message. The correct memory settings depend on your server hardware, the number of other applications, and the number of users on the server, and might be larger than these recommendations.

Database server requirements

Your database server must meet the minimum requirements listed in the *Platform Support Guide* on the Pega Community.

- Support for Java.
- Support for User-defined functions (UDFs) if you plan to use them.

Storage and logging requirements

Before you configure, configure your system to manage log storage space.

- Allocate enough storage to accommodate debugging and other logging requirements.
- Configure logging to avoid writing logs to the directory that contains the application server run-time components.

Configuring Java

Before you install, configure the JAVA_HOME environment variable.

1. Set JAVA_HOME to the root directory of the JDK.
2. Remove from the PATH any references to a Java shortcut.

Time zones, character encoding, and regional settings

Verify that your database server, application server, and the system on which you are deploying Pega Platform use the same:

- Time zone
- Character encoding (UNICODE or EBCDIC)
- Regional settings/locale

Database server configuration

Follow these instructions to prepare and configure your database server.

Prepare your database

Before you begin preparing your database, confirm that your database server is installed and running and verify that your database meets the minimum requirements.

- Verify that your system includes a supported version of the JDBC4 driver. Run the Stack Builder to download the JDBC driver.
- If you plan to use user-defined functions (UDF), enable Java in the database.
- Configure the PostgreSQL auto vacuum settings.

  Caution: To prevent transaction ID wraparound failures, PostgreSQL requires that you vacuum every table in every database at least once every two billion transactions, even when auto vacuum is enabled. When the threshold is reached, PostgreSQL sets the database to read-only and requires a lengthy manual vacuum. To avoid related data loss, configure your PostgreSQL database settings to dynamically monitor the transaction ID aging and schedule a vacuum freeze at a convenient time. For more information, see the PostgreSQL documentation.

Configuring your database

To prepare your database server for use with Pega Platform, complete the following steps:

1. Determine which database users you need and create database user accounts.
2. Create an empty database.
3. Create the database schema.

Database users

This section describes deployment and runtime users and lists all required permissions.

- Deployment user — This user performs actions only during the deployment.
- Run-time users — These users perform actions on the Pega Platform after the deployment. In a dual-user configuration, an Admin user is granted full privileges, and a Base user is granted a smaller subset. Pega recommends the dual-user configuration:
  - Base user — The user who runs the Pega Platform. Most run-time operations use the Base user and associated data source.

    Because PostgreSQL does not permit two users to have the same schema owner privileges, the Deployment user is the same as the Admin user.

  Pega recommends that you use the dual-user configuration with separate Admin and Base users; however, you can create a single Base user with both sets of privileges. If there is no separate Admin user, the Pega Platform uses the Base user for all run-time operations.

PostgreSQL user permissions

The permissions needed for your database users depend on whether you have a split-schema or a single-schema configuration, and whether you are using the recommended dual Admin/Base user configuration.

Because PostgreSQL does not permit two users to have the same schema owner privileges, the Deployment user is the same as the Admin user.

Split-schema configuration

	Base User *	Admin/Deployment User
Schemas owned by this user	None	*data-schemarules-schema*
Privileges	• USAGE ON SCHEMA *data-schema* • USAGE ON SCHEMA *rules-schema* • ALTER DEFAULT PRIVILEGES FOR USER *admin-user* IN SCHEMA *data-schema* GRANT ALL ON TABLES TO *base-user* • ALTER DEFAULT PRIVILEGES FOR USER *admin-user* IN SCHEMA *rules-schema* GRANT ALL ON TABLES TO *base-user* • IN SCHEMA *data-schema* GRANT EXECUTE ON FUNCTIONS • IN SCHEMA *rules-schema* GRANT EXECUTE ON FUNCTIONS	Because the Admin/Deployment user is the owner of the schema, they do not need any additional privileges.

* Pega recommends the dual-user configuration. For a single-user configuration, the Base user also requires the permissions listed for the Admin user.

Single schema configuration

	Base User *	Admin/Deployment User
Schema owned by this user	None	*schema-name*
Privileges	• USAGE ON SCHEMA *schema* • ALTER DEFAULT PRIVILEGES FOR USER *admin-user* IN SCHEMA *schema* GRANT ALL ON TABLES TO *base-user*	Because the Admin/Deployment user is the owner of the schema, they do not need any additional privileges.

* Pega recommends the dual-user configuration. For a single-user configuration, the Base user also requires the permissions listed for the Admin user.

Creating database users by using PostgreSQL tools

To create database accounts in PostgreSQL, either use the pgAdmin tool which provides a graphical interface or the PSQL tool which provides a command-line interface. The instructions for using pgAdmin are included in this guide. See the PostgreSQL documentation for information about PSQL.

- For dual-user configurations, create:
  - A Base user with read and write permission and the ability to run stored procedures and UDFs.
  - An Admin user with full access including permission to create and alter tables.
- If you want to use one account, create a user with the same authority and privileges as the Admin user described above.

To use the pgAdmin tool:

1. Launch the pgAdmin tool.
2. Double-click the PostgreSQL instance that you want to configure.

   Note: You must enter the password for the administrator account that was configured when PostgreSQL was installed.
3. Right-click **Login Roles** and select **New Login Role**.
4. In the **Role Name** field, enter the name for your database user.
5. On the **Definition** tab, enter the password for this user.
6. On the **Role Privileges** tab, grant permissions to this user. When using a single database user, you must grant at least the **Can create database objects permission**.
7. Click **OK**.
8. To add a second database user, repeat steps 3 - 7.

Create an empty database

Create a database with a minimum of 5 GB for the user tablespace and, if possible, allow the database to grow. This minimum size allows you to load the initial rulebase and do simple development. Monitor the database use carefully. As development begins, the size of the database will need to increase significantly, depending on the complexity of your Pega Platform applications and the number of users.

Creating an empty database by using PostgreSQL tools

Use the PostgreSQL tools to create an empty database. Make note of the access information to later configure the application server.

1. Right-click **Databases** and select **New Database**.
2. In the **Name** field, enter the name of the new database to create.
3. In the **Owner** drop down box, select the database user. For dual-user configurations, select the Admin user.
4. Optional. To shorten sort time in environments where you do not need language-specific sorting, set Collate = C:

   a) Open the **Definition** tab.

   b) In the **Collate** field, enter `C`.
5. Click **OK** to create the database.
6. Record the following database information required to access this database; you will need this information when you configure the application server.
   - Server name
   - Port number
   - Database name
   - Database user
   - Database password

PostgreSQL database schema

When using PostgreSQL, Pega Platform requires at least two schema: a schema for rules and data and a second schema for Java user-defined functions (UDFs).

Note: If you are using the PostGIS extension, it cannot be applied to either the rules or data schemas. If it is, you will be unable to upgrade your system in the future. PostGIS can be applied to any other schema.

Creating the primary schema

Create the primary schema for rules and data objects.

1. Optional: Enable remote access to the pgAdmin tool:
   a) Open the **`pg_hba.conf`** file in the data directory (**`/var/lib/pgsql/x.x.x_x/data`**).
   b) Add a line similar to the following to the end of the file:
   ```
   host all all 0.0.0.0/0 md5
   ```
   c) Save and close the file.
   d) Restart the PostgreSQL service.
2. In pgAdmin, click **+** to expand your database.
3. Under the database name, right-click **Schemas** and select **New Schema**.
4. On the **New Schema** window, in the **Name** field, enter the schema name in lowercase letters.

   Note: PostgreSQL requires that schema names are all lowercase. Enter the schema name only in lowercase letters without quotation marks, or the deployment will not work.
5. Use the **Owner** drop-down list to select your database user.
6. On the **Privileges** tab, select **ALL**, click **Add/Change** and select the privileges to grant to the database user.
7. Add the schema to the search path:
   a) In the **Databases** section, right-click your database name and select **Properties**.
   b) Select **Variables**.
   c) In the **Variable Name** field, select **search_path**.
   d) In the **Variable Value** field, enter the schema names.
      - For split schema systems, specify both the primary schema and the database schema:
        dataschema,rulesschema
      - For single schema systems, specify the rules schema name:
        rulesschema
   e) Click **Add/Change**.
   f) Click **OK** to add the variable.
   g) On the **Utility** screen, verify that the **Default schema property** correctly lists the new schema.

Installing PL/Java for Java user-defined functions

Pega Platform requires Java user-defined functions (UDFs). To work with PostgreSQL, UDFs require the PL/Java add-on for PostgreSQL and a new dedicated schema. The new schema, called SQLJ, is defined for the Pega Platform database.

For PostgreSQL 9.3 and later, the PL/Java module is included in the Pega Platform resource kit. Before you install the Pega Platform, install the version of PL/JAVA included in your resource kit. If you do not install and configure PL/Java before you attempt to install the Pega Platform, the Pega Platform installation fails. Follow the instructions for your platform.

Installing PL/Java on Windows

Follow these instructions to install the PL/Java module on Windows.

1. Install the Microsoft Visual C++ (MSVC) 2012 runtime integrated development environment from the Microsoft Windows site. Install MSVC even if it is already installed.
2. Restart your system.
3. Add %JRE_HOME%\bin\server to your PATH variable.
4. Locate the PL/Java library file in the %JRE_HOME%\bin\server directory. The library file will have one of the following names:
   - **jvm.dll**: This is the most common library file.
   - **libjvm.so**
   - **libjvm.dll**
   - **libjvm.dylib**
   - **jvm.so**
   - **jvm.dylib**
5. Copy the following files from **ResourceKit\PLJava\9.x\Windows** to the *PostgreSQL_Home*\lib64\ directory. If you are using the 32-bit version of PostgreSQL, your path may be different:
   - **pljava.dll**
   - **pljava.jar**
6. Configure the PostgreSQL virtual machine and library options:
   a) Open **postgresql.conf** in the PostgreSQL data directory and navigate to the end of the file.
   b) Enter a line similar to the following to add the classpath; use the double slash marks as shown:
   ```
   pljava.classpath='PostgreSQL_Home\\lib64\\pljava.jar'
   ```
   c) Optional. Enter a line similar to the following to set the virtual machine options:
   ```
   pljava.vmoptions = '-Xms32M -Xmx64M -XX:ParallelGCThreads=2'
   ```
   d) Enter the following line to set the working memory size:
   ```
   work_mem = 5MB
   ```
   e) At the end of the file, enter a line similar to the following to set the library location from step 4.
   ```
   pljava.libjvm_location = '%JRE_HOME%/bin/server/filename'
   ```
   f) Save and close **postgresql.conf**.
7. Stop and restart the PostgreSQL service.

Installing PL/Java on Linux

Follow these instructions to install the PL/Java module on Linux.

1. Locate the PL/Java library file in the **/usr/java/jdk-***X.X.X* **/jre/lib64/***system-name***/server** directory. The library file will have one of the following names:
   - **libjvm.so**
   - **libjvm.dll**
   - **libjvm.dylib**
   - **jvm.so**
   - **jvm.dll**
   - **jvm.dylib**
2. Create a symbolic link for libjvm.so in the /lib64 directory. Use a command similar to the following:
   `ln -s /usr/java/jdk`*X.X.X*`/jre/lib64/`*system-name*`/server/libjvm.so /lib/libjvm.so`
3. Copy the following files from **ResourceKit/PLJava/9.x/Linux** to the *PostgreSQL_Home***/lib64** directory. If you are using the 32-bit version of PostgreSQL, your path may be different:
   - **pljava.so**
   - **pljava.jar**
4. Configure the PostgreSQL virtual machine and library options:
   a) Open **postgresql.conf** in the PostgreSQL data directory and navigate to the end of the file.
   b) Enter a line similar to the following to add the classpath; use standard single Linux slash marks as shown:
   `pljava.classpath='`*PostgreSQL_Home*`/lib64/pljava.jar'`
   c) Optional. Enter a line similar to the following to set the virtual machine options:
   `pljava.vmoptions = '-Xms32M -Xmx64M -XX:ParallelGCThreads=2'`
   d) Enter the following line to set the working memory size:
   `work_mem = 5MB`
   e) At the end of the file, enter a line similar to the following to set the library location.
   `pljava.libjvm_location = '/usr/java/`*jdkx.x.x_x*`/jre/lib64/`*system-name*`/server/`*filename*`'`
   f) Save and close **postgresql.conf**.
5. Issue the following commands to stop and restart the PostgreSQL service:

```
service postgresql-x.x.x_x stop
service postgresql-x.x.x_x start
```

Creating secondary schema for Java user-defined functions

Use the PostgreSQL tool pgAdmin to create the new schema called sqlj with PL/Java correctly installed.

1. Open the **install.sql** file in the ResourceKit/PLJava/9. *x/OS* / directory in a text editor. You will copy the contents of this file into the PostgreSQL tool in a later step.
2. In pgAdmin, select the database in which you plan to install PL/Java.
3. Click the SQL icon.

4. In pgAdmin, paste the contents of the **`install.sql`** file into the pgAdmin editor.
5. Click **Execute pgScript**.
6. Verify that there are no error messages. If there is an error, check the library directory settings and restart the PostgreSQL service.

Pega Platform installation

There are multiple methods of installing Pega Platform.

- UI tool — The Installation and Upgrade Assistant is a Java-based UI tool that sets up the Pega Platform rules schema in the database and loads the Pega Platform rules.
- Command-line script — A command-line script automates the installation of Pega Platform in headless environments.

These methods use a batch process to load the rulebase. Because of the large number of rules and other data objects that must be loaded, Pega strongly encourages you to install on the same network as the database server. If this is not possible, install on a computer with fast, direct access to the database server. Do not attempt to install on a virtual private network (VPN) or a multi-hop wide area network (WAN).

Extracting and validating the distribution image

Follow these steps to extract and validate the distribution image:

1. Copy the compressed distribution image to the computer that you will use to run the installation. Extract the contents of the compressed file into an empty directory. If you are installing the software from a DVD, copy the contents of the DVD to an empty directory.
2. Verify the contents of the extracted distribution image.

   The **Pega-image\checksum** directory provides an MD5 checksum for each the file in the distribution image. To verify that the files downloaded and uncompressed correctly, calculate a checksum using the Jacksum tool at www.jonelo.de/java/jacksum/. For example, if you uncompressed the distribution image to **PEGA** enter the following command: `java -jar jacksum.jar -m -a md5 -r -p -O outputFile.md5`**PEGA**
3. Compare **outputFile.md5** to the md5 file located in **Pega-image\checksum**. The checksum values should be identical.

What to do next: Choose the installation method:

- To use the IUA, continue at Installing by using the Installation and Upgrade Assistant (IUA).
- To use the command line tool, continue at Editing the setupDatabase.properties file.

Installing by using the Installation and Upgrade Assistant (IUA)

Because of the large volume of data, run the IUA on the same network as the database server. If this is not possible, run the tool on a system with fast, direct access to the database server. The Deployment user performs these steps.

Pega Platform writes command-line output to a file in the **Pega-image\scripts\logs** directory.

The process can last for several hours and the time can vary widely based on network proximity to the database server.

To run the IUA:

1. Double-click the **PRPC_Setup.jar** file to start the IUA.

   Note: If JAR files are not associated with Java commands on your system, start the IUA from the command line. Navigate to the directory containing the **PRPC_Setup.jar** file, and type `java -jar PRPC_Setup.jar`.

   The IUA loads and the Pega icon is displayed in your task bar.
2. Click **Next** to display the license agreement.
3. Review the license agreement and click **Accept**.
4. On the **Installer Mode** screen, choose **Installation** and click **Next**.
5. Choose your database type and click **Next**.
6. Choose the product edition and click **Next**.
   - **Standard Edition** — Install a new instance of Pega 8.1.
   - **Multitenancy Edition** — Install Pega 8.1 in a specialized mode designed to support software-as-a-service (SaaS) run-time environments. For more information about multitenancy, see the *Pega Platform Multitenancy Administration Guide*.

   Note: Upgrading from one edition to another is not supported. The schema DDLs for the two editions are not compatible. For example, if you install the standard edition and later decide to use the multitenant edition, you must either drop and re-create the database or create a new database.
7. Configure the database connection. The JDBC drivers allow the Pega Platform application to communicate with the database.

   Note: Some of the fields on the **Database Connection** screen are pre-populated based on the type of database you selected. If you edit these or any other fields on this screen, and then later decide to change the database type, the IUA might not populate the fields correctly. If this occurs, enter the correct field values as documented below, or exit and rerun the IUA to select the intended database type.

   - **JDBC Driver Class Name** — Verify that the pre-populated value is accurate: `org.postgresql.Driver`
   - **JDBC Driver JAR Files** — Click **Select Jar** to browse to the appropriate driver files for your database type and version. For a list of supported drivers, see the *Platform Support Guide*.
   - **Database JDBC URL** — Verify that the pre-populated value is accurate.

     For information about URLs, see Obtaining database connection information.

     - To connect to PostgreSQL —

       `jdbc:postgresql://`*server*`:5432/`*database*
   - **Database Username and Password** — Enter the user name and password that you created for the Deployment user on your database.
   - **Rules Schema Name** — Enter the name of the rules schema in the database.
   - **Data Schema Name** — Enter the name of the data schema in the database. For single-schema configurations the data schema name is identical to the rules schema name.
   - **Customer Data Schema Name** — Optional: Enter the name of the customer data schema if it is separate from the data schema.
8. Click **Test Connection**. If the connection is not successful, review your connection information, correct any errors, and retest. When the connection is successful, click **Next**.

9. Optional: Specify whether you will have your database administrator manually apply the DDL changes to the schema. These changes include the user-defined functions (UDF) supplied by Pegasystems. By default, the tool generates and applies the schema changes to your database.
   - To generate and apply the DDL outside the UI tool, select **Bypass Automatic DDL Application** and continue the deployment. After you complete the deployment, manually generate and apply the DDL and UDF. For more information, see Optional: Generating and applying DDL and Optional: Installing user-defined functions.
   - To have the tool automatically apply the DDL changes and the UDF, clear **Bypass Automatic DDL Application**.
10. Enter the system name and production level and click **Next**:
    - **System Name** — Enter the name of your Pega Platform system. To find the system name, navigate to **System > Settings > System Name**.
    - **Production Level** — Enter a production level. The production level affects many security features of your system. Both the system name and production level can be changed after the system is running. Depending on the type of installation, choose:
      - `5` for a system that will be used in production
      - `4` for a preproduction system
      - `3` for a test system
      - `2` for a development system
      - `1` for an experimental system

      Edit the production level from the App Explorer. Enter `Data-Admin-System` in the search field and select **SysAdmin > Class > Data-Admin-System** to open your system.
11. Click **Start** to begin loading the rulebase.

    Logs display in the log window and are also stored in the **Pega-image** \scripts\logs directory. During the deployment, the log window might appear inactive when the IUA is processing larger files.
12. Click **Back** to return to the previous screen, and then click **Exit** to close the IUA.

What to do next: Determine the next step:

- If you opted to have the IUA automatically apply the schema changes, and you will not enable Kerberos authentication, configure the application server.
- If your database administrator will apply DDL manually, or if you will enable Kerberos authentication, continue at Editing the setupDatabase.properties file.

Editing the setupDatabase.properties file

Edit the **`setupDatabase.properties`** file to configure deployment scripts.

Skip this section if your deployment meets all the following criteria:

- You will use the Installation and Upgrade Assistant.
- You will allow the Installation and Upgrade Assistant to automatically apply the schema changes and do not need to create a DDL file.
- You will not enable Kerberos authentication.

If your deployment does not meet all these criteria, follow the steps in this section to edit the **setupDatabase.properties** file. The **setupDatabase.properties** file controls scripts which perform the following tasks:

- Install Pega Platform and enable Kerberos authentication. Use the **install.bat** or **install.sh** script.
- Generate a DDL file of schema changes. Use the **generateddl.bat** or **generateddl.sh** script. You can use the generateddl script regardless of whether you use the IUA or the command-line script.
- Generate user-defined functions. Use the **generateudf.bat** or **generateudf.sh** script.

1. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Directories.distributionDirectory*\scripts\setupDatabase.properties
2. Specify the properties for your system. For each property, add the appropriate value after the equal sign. See Database connection properties and script arguments.
3. Save and close the file.

Database connection properties and script arguments

The database connection properties in the **setupDatabase.properties** file specify the settings needed to connect to the database. The script arguments specify the same settings when you use command-line scripts. Command-line settings override property file settings.

Script argument	Property	Description
--driverJAR	pega.jdbc.driver.jar	Path and file name of the JDBC driver.
--driverClass	pega.jdbc.driver.class	Class of the JDBC driver
--dbType	pega.database.type	Database vendor type. Enter: `postgres`
--dbURL	pega.jdbc.url	The database JDBC URL. For more information, see Obtaining database connection information.
--dbUser	pega.jdbc.username	User name of the Deployment user.
--dbPassword	pega.jdbc.password	Password of the Deployment user. For encrypted passwords, leave this blank.
--adminPassword	pega.admin.password	For new installations only. The initial password for administrator@pega.com. If you do not set this password before you install, the installation fails.
	jdbc.custom.connection.properties	Optional: Semicolon-delimited list of custom JDBC properties. (for example: prop1=value;prop2=value;prop3=value)
--rulesSchema	rules.schema.name	In a single schema environment, sets rules schema and data schema. In a split-schema configuration, sets the rules schema only.
--dataSchema	data.schema.name	For split-schema configurations only, sets the data schema name.
--customerDataSchema	customerdata.schema.name	An optional customer data schema separate from the default Pega data schema.

Script argument	Property	Description
	user.temp.dir	Optional: The location of the temp directory. Set this location to any accessible location. For example, C:\TEMP.
--mtSystem	multitenant.system	Specifies whether this a multitenant system.

Optional: Enabling Kerberos authentication

Kerberos is a computer network authentication protocol that allows nodes communicating over a non-secure network to prove their identity to one another in a secure manner. Skip this section if you do not want to enable Kerberos authentication.

To enable Kerberos for authentication, you must use the command line to deploy Pega Platform:

1. Edit the setupDatabase.properties file.
   a) Open the **`setupDatabase.properties`** file in the scripts directory of your distribution image: *Directories.distributionDirectory*`\scripts\setupDatabase.properties`
   b) In the **Uncomment this property section** of the file, uncomment the custom property:
   ```
   jdbc.custom.connection.properties
   ```
   c) Provide the correct parameters as semicolon-delimited name/value pairs. The specific parameters depend on your security infrastructure, for example:
   ```
   jdbc.custom.connection.properties=
   parameter1=value1;
   parameter2=value2;
   parameter3=value3;
   ```
   d) Comment out all the user name and password properties so that they appear as follows:
   ```
   # pega.jdbc.username db username
   # pega.jdbc.password db password
   [lines removed here]
   # pega.jdbc.username=ADMIN
   # pega.jdbc.password=ADMIN
   ```
   e) Save and close the file.
2. Configure your database to enable Kerberos functionality. This might include additional vendor-specific JDBC driver configuration, or other setup procedures. For more information, see your database documentation.

What to do next: Continue at Installing from the command line.

Installing from the command line

Because of the large volume of data, run the command-line script on the same network as the database server. If this is not possible, run the script on a system with fast, direct access to the database server.

The **`install.bat`** and **`install.sh`** scripts use the properties in the **`setupDatabase.properties`** file. To overwrite any property, pass command line arguments.

1. If you have not done so already, edit the **`setupDatabase.properties`** file.
   a) Open the **`setupDatabase.properties`** file in the scripts directory of your distribution image: *Directories.distributionDirectory*`\scripts\setupDatabase.properties`
   b) Configure the connection properties. For more information about parameter values, see Properties file parameters.
   c) Set the initial administrator password. If you do not set this password before you install, the installation fails. The administrator must change this password after the first time they log in. For more information, see Logging in and changing the administrator password.
   `pega.admin.password=`*initial-admin-password*
   d) Save and close the file.
2. Open a command prompt and navigate to the scripts directory.
3. Type `install.bat` or `./install.sh` to run the script.

Installing the rulebase can take several hours, depending on the proximity of the database to the system running the installation script. When the installation is complete, you see a **BUILD SUCCESSFUL** message.

Pega Platform writes command-line output to a file in the **Pega-image**`\scripts\logs` directory.

What to do next: Now configure the application server.

Application server configuration

Follow the instructions in this section to configure your application server.

- Ensure that your application server meets the prerequisites listed in Application server requirements and in the *Platform Support Guide* on the Pega Community.
- Prepare and configure the application server.
- Deploy the Pega Platform applications.

Preparing to configure the application server

Complete these steps before you configure the application server:

1. Set your JAVA_HOME environment variable to point to the Java JDK directory.
   `JAVA_HOME=C:\Program Files\Java\jdkx.x.x_x\`
2. Ensure that your database JDBC jar file is accessible to Tomcat by saving it to the lib directory on your application server.
3. Ensure that your operating system references a common time standard such as the one available at www.time.gov.
   - On UNIX, this is the Network Time Protocol daemon, ntpd.
   - On Windows, you can set a similar service through the clock settings in the Windows Control Panel or task bar.

   See the documentation for your specific hardware operating system for information about setting this critical service.
4. Ensure that the following ports are open and available:
   - Search — One TCP port in the range 9300-9399 (the default is 9300). This port is used for internal node-to-node communication only, and should not be externally accessible.
   - Cluster communication — Leave open the port range 5701-5800. By default, the system begins with port 5701, and then looks for the next port in the sequence (5702, followed by 5703 and so on). To override the default port range, set a different value for the initialization/cluster/ports setting in the **`prconfig.xml`** file.

   Note: The number of available ports in this range must be greater than or equal to the greatest number of JVMs on any one node in the cluster. For example, if three JVMs are on one node, and seven JVMs on another node, at least seven ports must be available.
5. Obtain the following information from your database administrator to determine the database connection URL:
   - Host name
   - Port number
6. Add a headless AWT setting to the Java options in the configuration file that is passed to Java to allow the Pega Platform to render and display graphics:
   a) Open the configuration file that is passed to Java at startup. The specific file name and location is site-specific.
   b) Add the following JVM option:
   `-Djava.awt.headless=true`

c) Save and close the file.

What to do next: Continue at Data source resources, data source entries, and default schema entries.

Data source resources, data source entries, and default schema entries

The application server configuration defines the required data source resources, data source entries, and default schema entries:

- Data source resources — Data source resources define the Pega Platform database connection information. The number of data source resources depends on whether you have a single-user or dual-user configuration:
  - All systems require one data source resource for the Base user.
  - Dual-user configurations also require a second data source resource for the Admin user.
- Data source entries — Data source entries specify which data source resource to use for database operations in each schema. For dual-user environments, you must explicitly define two additional data source entries for the Admin user:
  - Admin data source entry for the rules schema
  - Admin data source entry for the data schema
- Default schema entries — Every system requires two entries that define the default schema names:
  - Default rules schema, for example, PegaRULES.
  - Default data schema, for example, PegaDATA. For single-schema configurations, the default data schema name is the same as the default rules schema name.

Continue at For Docker, multiple VMs, or multiple NICs: Setting the public address.

For Docker, multiple VMs, or multiple NICs: Setting the public address

If the cluster is set up in Docker, uses separate virtual machines (VMs), or multiple network interfaces (NICs), set the public address in the **prconfig.xml** file for each Pega Platform node.

1. Open the **prconfig.xml** configuration file in the prweb/WEB-INF/classes subdirectory of the application server directory. For more information, see the Pega Community article *How to change prconfig.xml file settings*.
2. Modify the **prconfig.xml** file. Add the following setting to set the public address:

   ```
   <env name=" identification/cluster/public/address" value=" IP address " />
   ```

   For example, if the IP address of the computer on which you run the Pega Platform node is 10.254.34.210, add the following setting:

   ```
   <env name=" identification/cluster/public/address" value="10.254.34.210" />
   ```

   The new setting controls the address that is used by the Pega Platform node.
3. Save and close the **prconfig.xml** file.
4. Repeat steps 1 to 3 for the remaining nodes.

What to do next: Continue at Configuring the application server.

Configuring the application server

These procedures describe a typical method for setting configuration properties in a basic installation.

Follow the steps in this section in order to configure your application server:

1. Validating database connections
2. Configuring Tomcat by editing the context.xml file
3. Configuring roles and users
4. Setting JVM parameters

Validating database connections

To avoid stale connections or closed connections being returned to the pool, work with your database administrator to validate the connections in your connection pool and configure the connection pool settings appropriately.

Configuring Tomcat by editing the `context.xml` file

Configure your Tomcat application server by editing the **`context.xml`** file.

1. Open the **`context.xml`** file and locate the <context> element. You will add all Pega Platform entries to the <context> element.
2. Add data source resources.
   a) Insert the following entry to add a data source resource for the Base user. Substitute your specific values for the host, port, database or service name, user name, password, and connection URL. For more information about connection URL strings, see Obtaining database connection information.

```
<Resource name="jdbc/PegaRULES"
auth="Container"
type="javax.sql.DataSource"









driverClassName="org.postgresql.Driver"
url="jdbc:postgresql://[host]:[port]/[dbname]"
username="Base user name"
password="password"



/>
```

   b) Configure the connection pool. Use this as an example. Determine the best value of this setting based on your application architecture, usage profile and environment considerations.

```
maxTotal="100"
maxIdle="30"
```

```
maxWaitMillis="10000"
```

c) Optional: For dual-user configurations, insert the following entry to add a data source resource for the Admin user.

```
<Resource name="jdbc/AdminPegaRULES"
auth="Container"
type="javax.sql.DataSource"









driverClassName="org.postgresql.Driver"
url="jdbc:postgresql://[host]:[port]/[dbname]
username="Admin user name"
password="password"




/>
```

d) Optional: For dual-user configurations, configure the connection pool information for the Admin user. Use this as an example. Determine the best value of this setting based on your application architecture, usage profile and environment considerations.

```
maxTotal="100"
maxIdle="30"
maxWaitMillis="10000"
```

3. For dual user configurations only, insert the following data source entry to specify the Admin data source resource for the data schema:

```
<Environment name="prconfig/database/databases/PegaDATA/dataSourceAdmin"
value="java:comp/env/jdbc/AdminPegaDATA" type="java.lang.String"/>
```

4. For dual user configurations only, insert the following data source entry to specify the Admin data source resource for the rules schema:

```
<Environment name="prconfig/database/databases/PegaRULES/dataSourceAdmin"
value="java:comp/env/jdbc/AdminPegaRULES" type="java.lang.String"/>
```

5. Insert the following entry to specify the rules schema name. Replace *RULES* with your rules schema name.

```
<Environment name="prconfig/database/databases/PegaRULES/defaultSchema"
value="RULES" type="java.lang.String"/>
```

6. Insert the following entry to specify the data schema name. Replace *DATA* with your data schema name.

```
<Environment name="prconfig/database/databases/PegaDATA/defaultSchema"
value="DATA" type="java.lang.String"/>
```

7. **Optional:** If your customer data schema is different than your Pega data schema, insert the following entry to specify the customer data schema name. Replace *customer-data-schema* with your customer data schema name.

```
<Environment name="prconfig/database/databases/CustomerData/defaultSchema"
value="customer-data-schema" type="java.lang.String"/>
```

8. Uncomment the Manager pathname parameter to turn off session persistence. When you are done, the parameter looks like the following example:

```
<!-- Uncomment this to disable session persistence across Tomcat restarts -->
<Manager pathname="" />
```

9. Insert the following entry anywhere in the body of the file to specify the temporary directory. Replace *path* with the entire path to your temporary directory:

```
<Environment name="url/initialization/explicittempdir" value="path"
type="java.lang.String"/>
```

Note: Each Pega Platform instance must have its own temporary directory.

10. Save and close the **context.xml** file.
11. Start the server to make these changes effective.

Configuring roles and users

Add the required roles, users, and passwords to the **tomcat-users.xml** file:

1. Open the **tomcat-users.xml** file in the *Tomcat_home* /conf/ directory.
2. Add the following lines anywhere in the file body:

```
          <role rolename="admin-gui"/>
          <role rolename="manager-gui"/>
          <user username="admin" password="admin" roles="admin-gui,manager-
gui,PegaDiagnosticUser"/>
```

3. Save and close the file.

Setting JVM parameters

Set the JVM memory options in the **setenv.bat** or **setenv.sh** file to increase the amount of system memory allocated to the application server running Pega Platform.

If you use either UNIX or Linux, set security to urandom.

1. Determine the memory parameter values. Pega recommends the following settings:
   - Initial Heap Size (Xms) — Between 4 GB - 8 GB, based on monitoring of memory usage and garbage collection frequency
   - Maximum Heap Size (Xmx) — Between 4 GB - 8 GB or larger, depending on your system configuration. For more information, see Application server requirements.

   If the server does not have enough memory allocated to run Pega Platform, the system can hang without an error message. The correct memory settings depend on your server hardware, the number of other applications, and the number of users on the server, and might be larger than these recommendations.
2. Set the memory parameters. The method depends on how you installed Tomcat:
   - If you installed Tomcat by expanding the compressed file:
     a. Create a **setenv.bat** file on Windows or a **setenv.sh** file in UNIX in Tomcat's bin directory.
     b. Use the following examples as a guide, and substitute your values for those given:

- On Windows — **setenv.bat**

  ```
  set JAVA_OPTS=-Xms4096m -Xmx8192m
  ```

- On UNIX — **setenv.sh**

  ```
  JAVA_OPTS="-Xms4096m -Xmx8192m"
  ```

- If you installed Tomcat as a Windows service, use the Tomcat utility to set the JVM options. Enter a command similar to the following.

  ```
  tomcat7.exe //US//Tomcat7 ++JvmOptions=-Xms4096m -Xmx8192m
  ```

3. If you use either UNIX or Linux, enter the following argument to set security to urandom:`-Djava.security.egd=file:///dev/urandom`
4. Set the node type with the JVM argument: `-DNodeType=`<node type>.

   To support queue processing, Pega Platform requires at least one stream node (node type `Universal` or `Stream`). In a single node cluster, set the node type to `Universal`, otherwise set the node type to `Stream`; for example, `-DNodeType=Stream`.

   If you have a high availability environment, configure at least two stream nodes by using the JVM argument: `-DNodeType=Stream`.
5. Save and close the configuration file.

Deploying the Pega Platform WAR or EAR file

After you configure your application server, deploy the **prweb.war** .

Note: When the server restarts after the application deploys, the first node you bring up becomes the default search node.

Deploying Pega Platform

1. Copy the **prweb.war** file from the **Pega-image\archives** directory to the *Tomcat_home***webapps** directory.
2. Restart the application server.
3. Shut down the server and delete the **prweb.war** file from the *Tomcat_home***webapps** directory to prevent Tomcat from redeploying the application each time the server restarts.

Post-deployment configuration

This section describes the post-deployment activities that are performed in the system after you have completed the setup and configuration of your application server and deployed the archives.

- Starting applications
- Logging in to the Pega Platform and changing your password
- Configuring Directed Web Access
- Configuring search index host node settings
- Changing the default path to the logs
- Log file size
- Database size
- Installing custom applications
- Enabling server-side screen captures for application documents
- Enabling operators

Starting Pega Platform

Ensure that the application server is running and start prweb.

Logging in and changing the administrator password

To test the deployment and index the rules, log in to Pega Platform web application. For security, you must change the administrator password.

1. Navigate to the PRServlet URL, replacing the *server* and *port* values with your specific values.
   `http://server:port/prweb`
2. Use the following credentials to log in the first time:
   - User ID — `administrator@pega.com`
   - Password — the password you set when you installed

   After logging in, Pega Platform indexes the rules in the system to support full-text search. During the index process, there might be a delay in the responsiveness of Pega Platform user interface. The process usually takes from 10 to 15 minutes to complete depending on your system configuration.

   If the index process ends without generating an error message, the deployment is successful.
3. Immediately after the index process completes, change the administrator password. Because the default password is widely published, it is important to protect your system after an installation by changing the password. The new password must be at least 10 characters long.

   If the system does not prompt you to change your password, follow these steps:

   a) From the **Operator Menu**, select the **Profile**.

   b) Click **Change Password**.

   c) Verify the **Current Password**, and then enter and confirm the **New Password**.

   d) Click **Save**.

Configuring Directed Web Access

A Directed Web Access (DWA) address allows you to grant one-time access to external users to enable them to process an assignment in your application. When you grant the access, the Pega Platform sends an email to the external user; this email includes a URL to access the application and can identify a proxy server.

Follow these instructions to configure the URL:

1. In the header of Dev Studio, click **Configure** > **System** > **Settings** > **URLs**.
2. In the **Public Link URL** field, enter the URL that you want to provide in emails in this format:
   `http://`*host:port*`/prweb`
3. Click **Save**.
4. Log out and log back in to Dev Studio for these changes to take effect.

Configuring search index host node settings

The Pega Platform supports full-text search for rules, data instances, and work objects. By default, search indexing is enabled and indexing starts when you start the application server after deploying the Pega Platform. The first node that starts after the deployment becomes the default initial search node. The default index directory is PegaSearchIndex in your temporary directory.

After the search indexes are completely built, you can change the default settings. Do not stop or bring down the default node until the search indexes build completely. The Search Landing Page displays the status.

Follow these steps to configure the search index host node settings:

1. Check your directory sizes. Ensure that the directories for all Elasticsearch host nodes have sufficient free space to hold the Elasticsearch indexes.
2. In the header of Dev Studio, click **Configure** > **System** > **Settings** > **Search**.
3. Expand **Search Index Host Node Setting**.
4. Specify one node to set as the Host Node. If necessary, delete all but one node. This is the node on which Elasticsearch indexes will be built.

   Note: Do not include more than one node in the **Search Index Host Node Setting** list. Including more than one node in the list at this point might cause duplicate index builds and compromise system performance. You will create additional nodes later in this process.
5. Verify the **Search Index Host Node ID** and the **Search Index File Directory**.
6. Expand **Automated Search Alerts**, and enable **Automatically Monitor Files**.
7. Click **Submit** to save the settings.
8. After the first indexing is complete, add any needed additional host nodes. The system replicates the indexes on the new nodes.

   Note:

   - Configure a minimum of two Elasticsearch host nodes. Pegasystems recommends that you configure a minimum of three nodes for maximum fault tolerance. You might need more than three nodes depending on the size of your cluster.
   - You can specify that a node is either always an index host node or that it never becomes an index host node even if it is the first node that is started after installation using the –

`Dindex.directory` JVM setting. To specify that a node is always an index host node specify the directory name. To specify that a node is never an index host node, leave this setting blank. If this setting is not used and a custom index file directory is specified on the Search landing page, the system uses the default directory when no other index host node is online when a server starts. For more information about configuring index host nodes, see Managing Elasticsearch index host nodes outside of the Search landing page on the Pega Community.

9. To enable communication between Elasticsearch host nodes in the cluster, open a TCP port in the range 9300-9399 on each node. (By default, Elasticsearch uses port 9300.) These ports are used for internal node-to-node communication only, and should not be externally accessible. Ensure that these ports are not subject to an idle connection timeout policy in the software or hardware that runs between these host nodes.

Changing the default path to logs

Changing the default path to Pega Platform logs

The default path for Pega Platform logs is the work\Catalina directory, for example:

`<`*Tomcat-home*`>\work\Catalina\localhost\prweb`

To avoid losing logging files if the default directory is overwritten, edit the **`prlog4j2.xml`** file to change the logging file directory:

1. Identify the directory you want to use for logging. If the directory does not yet exist, create the directory. The Tomcat user must have write permissions for the new directory.
2. Back up the existing **`prlog4j2.xml`** file in the Pega-image\scripts\config directory.
3. Open the **`prlog4j2.xml`** file for editing.
4. Locate the appender for the **`PegaRules.log`** file that contains the `${sys:pega.tmpdir}` string:

   ```
   <RollingRandomAccessFile name="RollingRandomAccessFile"
   fileName="${sys:pega.tmpdir}/PegaRULES.log"...
   ```

5. Change the fileName parameter to use the new directory:

   `fileName="`*log-directory*`\PegaRULES.log"`

   Note: The directory path is relative to the < *Tomcat-home* >\work directory. In this example, the log files will be created in < *Tomcat-home* >\work\log-directory.

6. Repeat steps 4 and 5 for the **`PegaRULES-ALERT.log`** and **`PegaRULES-ALERTSECURITY`** files.
7. Save and close the **`prlog4j2.xml`** file.
8. Restart the Pega Platform to make the changes.
9. Check the directory to confirm that the log files were created in the new location.

Log file size

The initial log file size is 50 MB. After the installation is complete, increase the size of the log files. The specific size will depend on your environment and the size of your application. For more information, see the Support area on the Pega Community.

Database size

Monitor the database use carefully. As development begins, the size of the database will need to increase significantly, depending on the complexity of your Pega Platform applications and the number of users.

Install applications

Install any applications now. If you obtained your application from Pega, follow the instructions in the Installation Guide for your application.

Caution: Grant the database user permissions as described in Database users. Some applications use triggers. During startup, Pega Platform checks for triggers that reference the upgrade cache and rule view tables; if these triggers exist, Pega Platform attempts to drop them. If the user does not have the correct permissions, Pega Platform cannot drop the triggers and fails to start up.

If you installed the applications before you deployed Pega Platform, Pega Platform automatically drops the triggers and this error does not occur.

Enabling server-side screen captures for application documents

Regardless of which application server platform you use, you must set up a Tomcat server to support taking and storing screen captures on a server rather than on a client. By taking and storing screen captures on a server, you avoid client-side limitations, such as browser incompatibilities or client software requirements.

As a best practice, virtually install Tomcat and deploy the **`prScreenShot.war`** file on the same server that is running Pega Platform. Otherwise, use a standalone Linux or Windows server. If you use a Linux server, you must include the following components:

- fontconfig
- freetype
- libfreetype.so.6
- libfontconfig.so.1
- libstdc++.so.6

You can include screen captures in an application document that is generated by the Document Application tool. Screen captures provide stakeholders with a realistic picture of an application's user interface. Install a PhantomJS REST server to include screen captures in an application document.

1. Download the following WAR file: **`Pega_DistributionImage\Additional_Products\PhantomJS\prScreenShot.war`**
2. Deploy the WAR file on a Tomcat server.

3. Edit the **tomcat-users.xml** file to add the following role and user. This file is located at \apache-tomcat-XX\conf\ tomcat-users.xml.

   ```
   <role rolename="pegascreencapture" /> <user username="restUser" password="rules" roles="pegascreencapture" />
   ```

4. Start the Tomcat server. The service is hosted at http://*IPaddress*:*port*/prScreenShot/rest/capture, where *IPaddress* is the address of the system where Tomcat is hosted, and *port* is the port on which the service is deployed.
5. Log in to your Pega Platform application and make the following changes:
   a) Edit the Data-Admin-System-Setting instance *Pega-AppDefinition* - *CaptureScreenshotsResourcePath* with the URL of the service, for example, `http://10.224.232.91:8080/prScreenShot/rest/capture`.
   b) Add the user that you created in step 3 to the Data-Admin-Security-Authentication profile instance CaptureScreenshotsAuthProfile.

What to do next: Continue at Configuring PhantomJS REST server security for including screen captures in an application document.

Configuring PhantomJS REST server security for including screen captures in an application document

To ensure a secure installation of Pega Platform, enable the PhantomJS REST server to take and store server-side screen captures. In application documents generated by the Document Application tool, screen captures provide stakeholders with a realistic picture of the application's user interface.

1. Obtain the SSL certificate from the Pega Platform administrator.
2. Add the SSL certificate to the list of trusted certificates:
   a) Double-click the certificate.
   b) Click **Install certificate** to start the **Certificate Import** wizard.
   c) Click **Next**, and select **Place all certificates in the following store**.
   d) Click **Browse**, select **Trusted Root certificate**, and click **OK**.
   e) Click **Next**, and then click **Finish** to complete the wizard.
3. Add the certificate to the truststore of the JVM on which the REST server is installed:
   a) Open a command prompt.
   b) Change the root directory to the security folder in the Java installation folder. For example, `C:\Program Files (x86)\Java\jre7\lib\security`.
   c) Run the following command:
   `keytool -keystore cacerts -importcert -alias` *certificate alias* `-file` *certificate name*
   d) When prompted, enter the password for the cacerts keystore. The default password is `changeit`.

Enabling operators

Pega Platform deployment security requires an administrator to enable new operators shipped with Pega Platform and requires password changes after the first login.

The administrator and new operators shipped with Pega Platform must change their passwords when they first log in:

- Batch@pega.com
- DatabaseAdmin@pega.com
- ExternalInviteUser
- IntSampleUser
- PRPC_SOAPOper
- PortalUser@pega.com
- UVUser@pega.com
- External

For more information about changing the administrator password, see Logging in and changing the administrator password.

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Authentication** > **Operator Access**.
2. In the **Disabled operators** list, click the link for the Pega-provided operator that you want to enable. The following standard operators are installed but disabled by default. When these standard operators first log on, they are required to change their passwords. Enable only those operators you plan to use:
   - Batch@pega.com
   - DatabaseAdmin@pega.com
   - ExternalInviteUser
   - IntSampleUser
   - PRPC_SOAPOper
   - PortalUser@pega.com
   - UVUser@pega.com
   - External
3. On the **Edit Operator ID** page, on the Security tab, select **Force password change on next login** and clear **Disable Operator**.
4. Select **Update password**.
5. Enter a password that conforms to your site standards and click **Submit**.
6. Click **Save** and close the operator page.
7. Repeat steps 2 through 6 for the remaining operators.

Appendix A — Properties files

The Pega Platform properties files include several database-specific properties.

- JDBC driver JAR file:

  postgresql-42.2.2.jar
- Database driver class — org.postgresql.Driver
- Database vendor type — postgres
- JDBC URL — url="jdbc:postgresql:// *host:port/dbname* "

Appendix B — Troubleshooting

Use the information in this section to troubleshoot errors. The error logs are displayed in the Installation and Upgrade Assistant window and are also stored in the **Pega-image** \scripts\logs directory.

Recovering from a failed deployment

If the deployment fails, follow these steps to drop the schemas and start a new installation:

1. Review the log files in the \scripts\logs directory.
2. Make any necessary changes to your system. If the error was due to a data entry mistake, make note of the correct information.
3. Generate the DDL files and drop the schemas:
   a) Verify the settings in the **setupDatabase.properties** file. For information about the properties, see Editing the setupDatabase.properties file.
   b) At a command prompt, navigate to the **Pega-image** \scripts directory.
   c) Run the **generateddl.bat** or **generateddl.sh** script with the `--action=drops` option, for example:
   ```
   generateddl.bat --action=drops
   ```
   d) Review the DDL files in the **Pega-image** \schema\generated\ output directory.
   e) Have your database administrator apply the DDL to drop the schemas.
4. Repeat the installation steps.

PEGA0055 alert — clocks not synchronized between nodes

The Pega Platform validates time synchronization to ensure proper operations and displays a PEGA0055 alert if clocks are not synchronized between nodes.

For information about how to reference a common time standard, see the documentation for your operating system.

ClassNotFoundException error — session persistence

During application server shutdown, Tomcat persists session information into the **session.ser** file in the server file directory. When the application server restarts, it reloads the session information from the **session.ser** file and deletes the file. If serialized session objects refer to classes that are not visible to the container layer, you see a ClassNotFoundException error.

This is a sample error message:

```
May 19, 2016 2:37:46 PM org.apache.catalina.session.StandardManager
 doLoad SEVERE: ClassNotFoundException while loading persisted sessions:
 java.lang.ClassNotFoundExceptioncom.pega.pegarules.session.internal.authorization.ContextM
    java.lang.ClassNotFoundException:
 com.pega.pegarules.session.internal.authorization.ContextMapDiagCallback
```

To suppress these errors, turn off Tomcat session persistence in the **`context.xml`** file.

System hangs with no error message — insufficient memory

If the server does not have enough memory allocated to run the Pega Platform, the system can hang without an error message. The correct memory settings depend on your server hardware, the number of other applications, and the number of users on the server, and might be larger than the minimum recommendations in System requirements.

Obtain database connection information

Before you configure the data source resources, obtain the correct database connection information from your database administrator.

To determine the database connection URL, obtain the following information from your database administrator:

- Host name
- Port number

When you configure the application server, you will enter the connection string, **pega.jdbc.url** as follows. Replace items in *italics* with the values for your system:

`jdbc:postgresql://`*server*`:`*port*`/`*database*

Optional: Generating and applying DDL

If you opted not to have the Installation and Upgrade Assistant automatically apply the DDL, generate and apply the DDL manually.

Generating the DDL file

Follow these steps to generate a DDL file for your database administrator to apply manually.

1. Edit the **setupDatabase.properties** file.
   a) Configure the connection properties. For more information about parameter values, see Properties file parameters. The customer data schema is optional.

```
# Connection Information
pega.jdbc.driver.jar=\path-to-the-database-JAR-file\DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment username
pega.jdbc.password=password
rules.schema.name=rules-schema-name
data.schema.name=data-schema-name
customerdata.schema.name=optional-customer-data-schema
```

   b) Save and close the file.
2. At a command prompt, navigate to the **Pega-image** \scripts directory.
3. Run **generateddl.bat** or **generateddl.sh** and pass in the required --action argument:

```
#generateddl.bat --action install
```

If you do not specify an output directory, the script writes the output to the default directory: **Pega-image\schema\generated**

Note: The output directory is deleted and re-created each time the generateddl script runs. To save a copy of the DDL, rename the directory before you run the script.

Applying the DDL file

Before you continue, have your database administrator follow these general steps to apply the schema changes; these schema changes can include changes to user-defined functions:

1. Review the DDL file in the output directory and make any necessary changes.
   The default directory is: **Pega-image\schema\generated\database\postgres**
2. Apply the DDL file.
   a) Register the Java .jar file with the database.
   b) Apply the **CREATE FUNCTION** DDL.

The output directory is deleted and re-created each time the generateddl script runs. To save a copy of the DDL, rename the directory before you rerun the script.

Editing the setupDatabase.properties file to bypass DDL generation

After your database administrator applies the changes to your database, configure the **setupDatabase.properties** file to bypass applying a schema that already exists. Reapplying an existing schema would cause the deployment to fail.

1. Open the **setupDatabase.properties** file in the scripts directory of your distribution image: *Directories.distributionDirectory***\scripts\setupDatabase.properties**
2. Set the property `bypass.pega.schema=true`.
3. Save and close the file.

Installing user-defined functions

The user-defined functions (UDFs) enable the Pega Platform to read data directly from the BLOB without creating and exposing columns. Skip this section if you installed the UDFs when you deployed Pega Platform.

There are several ways you might have bypassed generating and installing the UDFs when you deployed:

- Setting either `bypass.pega.schema=true` or `bypass.udf.generation=true` in the **`setupDatabase.properties`** file
- Setting `pega.target.bypass.udf=true` in the **`migrateSystem.properties`** file
- Selecting **Bypass Automatic DDL Application** from the Installation and Upgrade Assistant

Before you install the UDFs, verify that you have the appropriate user permissions.

For more information about user permissions, see Database users.

1. Edit the **`setupDatabase.properties`** file.
   a) Configure the connection properties. For more information about parameter values, see Properties file parameters.

```
# Connection Informationpega.jdbc.driver.jar=\path-to-the-database-JAR-file\DRIVER.jar
pega.jdbc.driver.class=database driver class
pega.database.type=database vendor type
pega.jdbc.url=URL of the database
pega.jdbc.username=Deployment user name
pega.jdbc.password=password
rules.schema.name= rules-schema-name
data.schema.name=data-schema-name
```

   b) Save and close the file.
2. On the rules schema, run the following commands to remove any partially installed UDFs:

```
DROP FUNCTION rules-schema-name.pr_read_from_stream;
DROP FUNCTION rules-schema-name.pr_read_decimal_from_stream;
DROP FUNCTION rules-schema-name.pr_read_int_from_stream;
```

3. Optional: If you have a split-schema, on the data schema run the following commands:

```
DROP FUNCTION data-schema-name.pr_read_from_stream;
DROP FUNCTION data-schema-name.pr_read_decimal_from_stream;
DROP FUNCTION data-schema-name.pr_read_int_from_stream;
```

4. From the **Pega-image** \scripts directory, run the **`generateudf.bat`** or **`generateudf.sh`** script with the `--action install` argument.