Installing ThingWorx 7.3

Version 1.2

Document Revision History

Revision Date	Version	Description of Change
January 26, 2017	1.2	Updated copyright information, added clarification for RHEL install and default cache setting.
January 03, 2017	1.1	Updated optional password encryption info.
December 15, 2016	1.0	Initial version for 7.3

Installing ThingWorx

Installing ThingWorx Core

ThingWorx Core is currently supported on Windows, Ubuntu, Amazon EC2, and Red Hat Enterprise Linux.

Prerequisites

Prerequisite software includes Apache Tomcat and Oracle Java. PostgreSQL is also required if you are not using H2 or SAP HANA. If you are installing ThingWorx for the first time, this document provides step-by-step installation instructions for your environment.

Upgrading

If you are upgrading to a newer version, refer to the Upgrading ThingWorx guide.

Database Options: PostgreSQL, SAP HANA, or H2

With ThingWorx 7.3, you can use PostgreSQL (with an optional High Availability layer), SAP HANA, or H2 for your data solution. If you are upgrading to 7.3, the following download package options are available when obtaining the thingworx.war from PTC Software Downloads:

- H2: **Thingworx-Platform-H2-7.3.0**
- PostgreSQL/HA: **Thingworx-Platform-Postgres-7.3.0**
- SAP HANA: **ThingWorx-Platform-Hana-7.3.0**

  NOTE: If you are using SAP HANA as your database option, refer to the Getting Started with SAP HANA and ThingWorx Guide for all installation and configuration details.

High Availability Option

With ThingWorx 7.0 and later, you can use PostgreSQL with an optional High Availability layer at the database level and/or at the ThingWorx level. Additional steps for HA are required and are located in the ThingWorx High Availability Administrator's Guide.

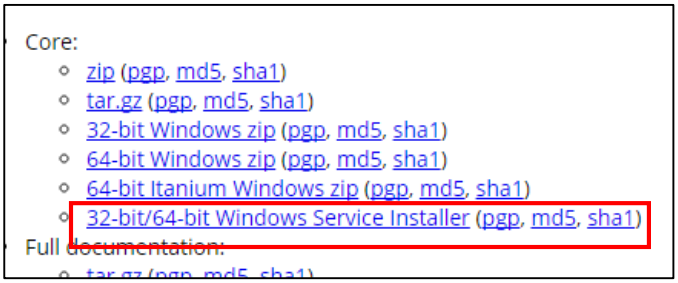
For detailed software and hardware requirements, refer to the ThingWorx System Requirements and Compatibility Matrix document.

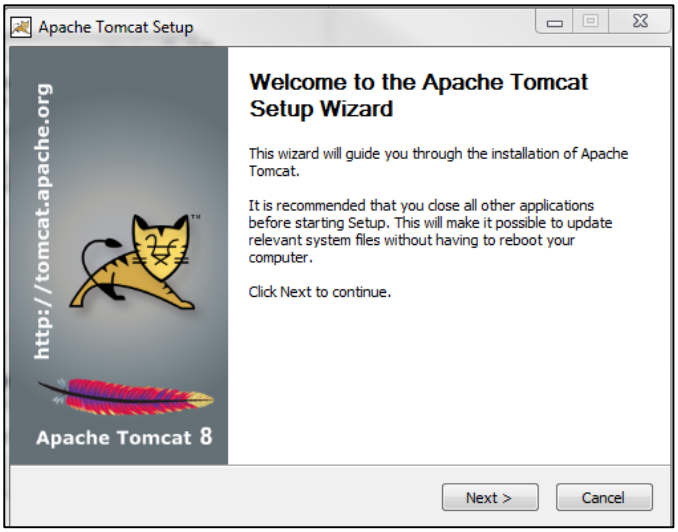
Installing ThingWorx for the First Time: H2 or PostgreSQL on Windows

NOTE: If you are using PostgreSQL or SAP HANA for your database, additional steps are required. If you are installing H2, steps are included below to skip all PostgreSQL sections.

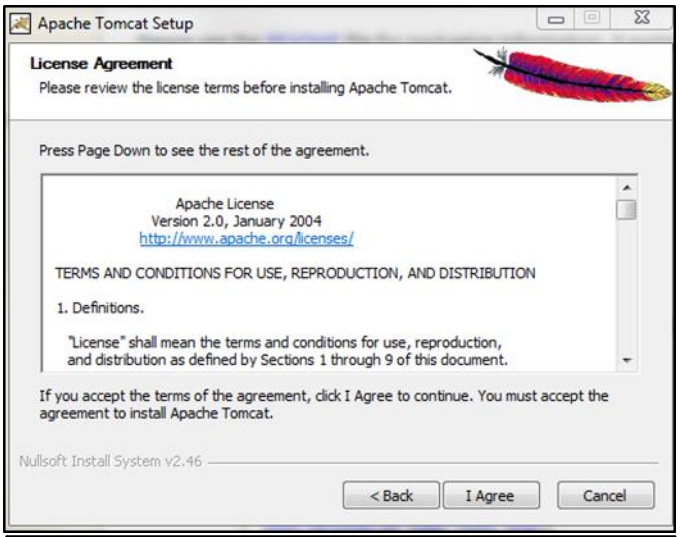
Installing Oracle Java and Apache Tomcat (Windows)

1. Download and install the required version of Java from the Oracle website.
   NOTE: Refer to the System Requirements and Compatibility Matrix document for version requirements.
2. Visit the Tomcat website to download the **32-bit/64-bit Windows Service Installer (pgp, md5, sha1).**
   NOTE: Refer to the System Requirements and Compatibility Matrix document for version requirements.
3. The Apache Tomcat Setup Wizard launches. Click **Next**.

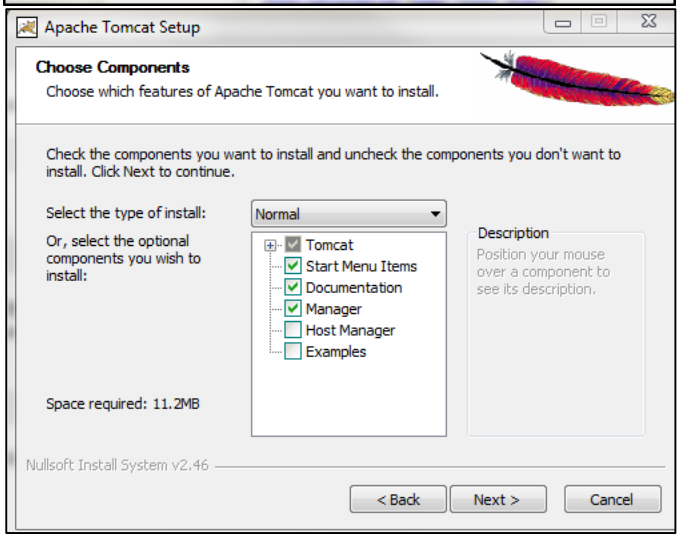




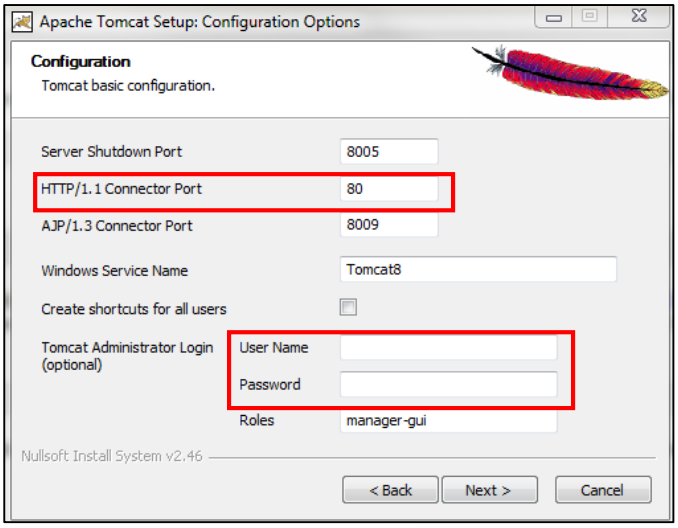
4. Click **I Agree**.



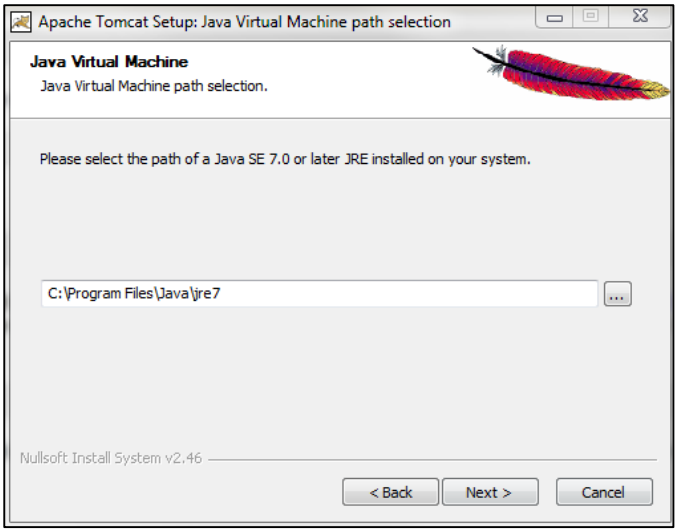
5. In the **Components** section, use the default settings.
6. Click **Next**.



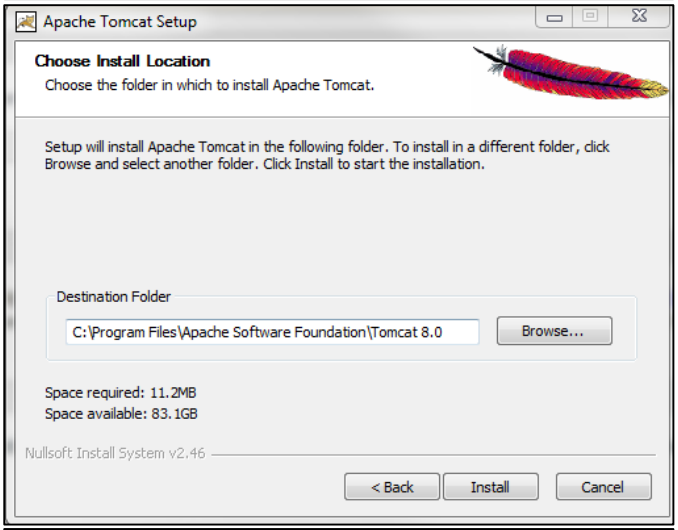
7. In the **HTTP/1.1 Connector Port** field, type **80** (or other available port).
8. In the **Tomcat Administrator Login** fields, type a **User Name** and **Password**.
9. Click **Next**.



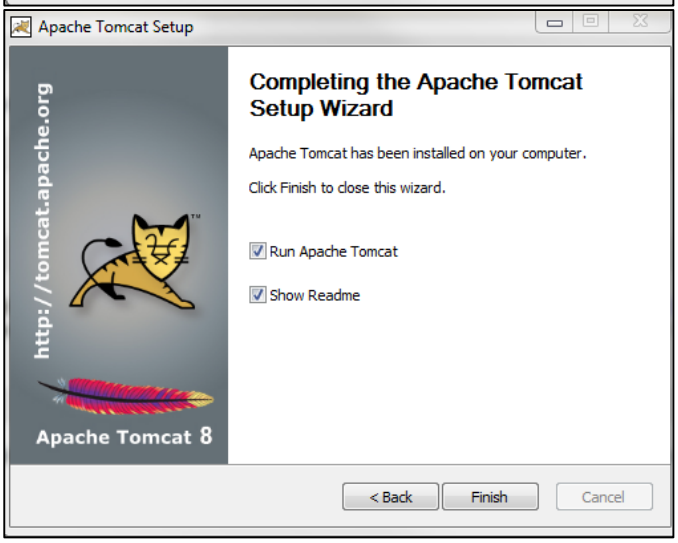
10. Enter the path to the proper 64-bit Java installation directory.
11. Click **Next**.



12. Click **Install**.

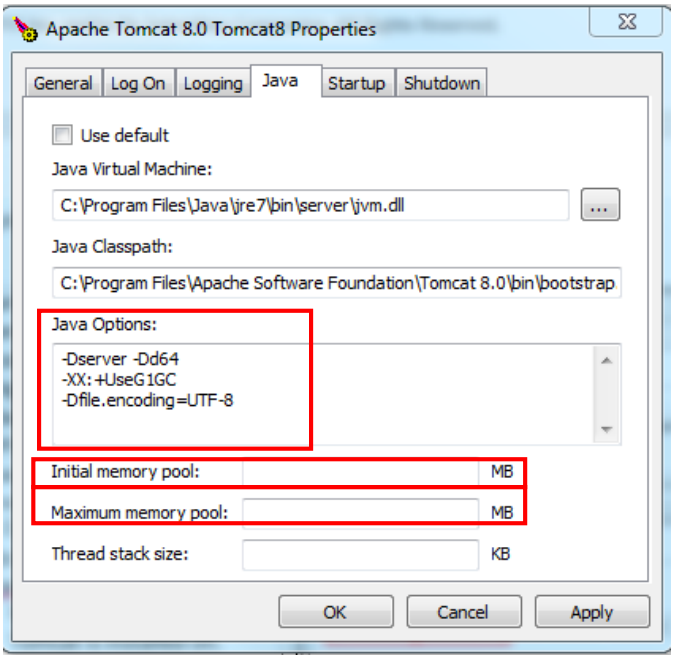


13. Click **Finish**.



14. Click **Start>Configure Tomcat**.
15. Open the **Java** tab.
16. In the Java Options field, add the following to the end of the options field:
    ```
    -Dserver -Dd64
    -XX:+UseG1GC
    -Dfile.encoding=UTF-8
    ```

    NOTE: For more information on these options and for additional options for hosted and/or public-facing environments, refer to the Appendix: Tomcat Java Option Settings.
17. Clear any values in the **Initial memory pool** and **Maximum memory pool** fields.
18. Click **OK**.



19. Go to the location of the Tomcat installation and open the **server.xml** file in the **conf** folder. For example, C:\Program Files\Apache Software Foundation\Tomcat 8.0\conf\server.xml
    Replace `HTTP/1.1` with `protocol="org.apache.coyote.http11.Http11NioProtocol"`

```
    <!--
    <Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true" scheme="https"
secure="true"
               clientAuth="false" sslProtocol="TLS" />
    -->
```

20. Save and close the file.
21. OPTIONAL STEP: If you want to increase the default cache settings that affect static file caching, add the following line within the <context></context> tags in the $TOMCAT_HOME/conf/context.xml file:
    ```
    <Resources
    cacheMaxSize="501200"
    cacheObjectMaxSize="2048"
    cacheTtl="60000"/>
    ```

- NOTE: Increasing this setting improves performance and avoids the following message in Tomcat:

  ```
  WARNING: Unable to add the resource
  at [/Common/jquery/jquery-ui.js] to
  the cache because there was
  insufficient free space available
  after evicting expired cache
  entries - consider increasing the
  maximum size of the cache
  ```

22. If you are installing H2, skip to the Installing ThingWorx section.

    NOTE: If you are using SAP HANA as your database, refer to the Getting Started with SAP HANA and ThingWorx Guide.

Installing and Configuring PostgreSQL (Windows)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers).

NOTE: If you are including the HA layer to your implementation, refer to the ThingWorx High Availability Administrator's Guide.

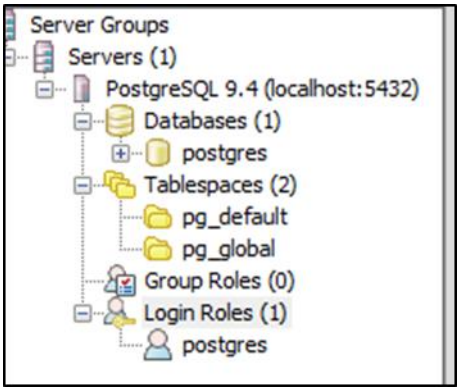
Installing PostgreSQL and Creating a New User Role in PostgreSQL (Windows)

1. Download and install the appropriate version of PostgreSQL from the following site: http://www.postgresql.org/download/

- pgAdmin III Tool
  - PgAdmin III is an open source management tool for your databases that is included in the PostgreSQL download. The tool features full Unicode support, fast, multithreaded query, and data editing tools and support for all PostgreSQL object types.

2. Open PostgreSQL using pgAdmin III.

3. Create a new user role (in this example, it is **twadmin**):
   a. Right click **PostgreSQL9.4** (localhost:5432).
   b. Select **NewObject>New Login Role**. On the **Properties** tab, in the **Role name** field, type **twadmin**.
   c. On the **Definition** tab, in the **Password** field, type **password** (must type twice).
4. Click **OK**.
   NOTE: Remember the user role name created in this step for later use.



Configuring and Executing the PostgreSQL Database Script (Windows)
To set up the PostgreSQL database and tablespace, the **thingworxPostgresDBSetup.bat** script must be configured and executed.

1. Add the **<postgres-installation>/bin** folder to your system path variable.
2. Create a directory named **ThingworxPostgresqlStorage** on the drive that Thingworx Storageis located (in the root directory by default).

   NOTE: If you create with the -**d<databasename>** command, you do not have to use the postgres user.

   NOTE: You must specify the **-l** option to a path that exists. For example, **-l D:\ThingworxPostgresqlStorage**. The script does not create the folder for you.

   The folder must have appropriate ownership and access rights. The folder should be owned by the same user who runs the PostgreSQL service, and have Full Control assigned to that user - this user is generally **NETWORK_SERVICE**, but may differ in your environment.

3. Obtain and open **thingworxPostgresDBSetup.bat** from the ThingWorx software download package.
4. Configure the script. Reference the configuration options in the table below.

   Various parameters such as **server**, **port**, **database**, **tablespace**, **tablespace location** and **thingworxusername** can be configured in the script, depending on your requirements. Execute this script with the **--help** option for usage information.

   As an example, to set up the database and tablespace with a default Postgres installation that has a postgres database as well as a postgres user name and assuming the user created above is **twadmin**, enter:

   ```
   thingworxpostgresDBSetup –a postgres –u twadmin –l
   C:\ThingworxPostgresqlStorage
   ```

   where **twadmin** is the user name

5. Execute the script. Once executed, this creates a new database and tablespace with a default PostgreSQL installation in the PostgreSQL instance installed on the localhost.

   NOTE: You may need to run the command prompt as admin.

thingworxPostgresDBSetup.bat Script Options

Option	Parameter	Default	Description	Example
-t or -T	server	localhost	Tablespace name	-t thingworx

Option	Parameter	Default	Description	Example
-p or -P	port	5432	Port number of PostgreSQL	-p 5432
-d or -D	database	thingworx	PostgreSQL Database name to create	-d thingworx
-h or -H	tablespace	thingworx	Name of the PostgreSQL tablespace.	-h localhost
-l or -L	tablespace_location	/ThingworxPostgresqlStorage	Required. Location in the file system where the files representing database objects are stored.	-l or -L
-a or -A	adminusername	postgres	Administrator Name	-a postgres
-u or -U	thingworxusername	twadmin	User name that has permissions to write to the database.	-u twadmin

Configuring and Executing the Model/Data Provider Schema Script (Windows)

To set up the PostgreSQL model/data provider schema, the **thingworxPostgresSchemaSetup.bat** script must be configured and executed. This will set up the public schema under your database on the PostgreSQL instance installed on the localhost.

1. Obtain and open the **thingworxPostgresSchemaSetup.bat** from the ThingWorx software download package.
2. Configure the script. Reference the configuration options in the table below.

   Various parameters such as **server**, **port**, **database**, **username**, **schema**, and **option** can be configured in the script depending on the requirements. Execute this script with **--help** option for usage information.

3. Execute the script.
   NOTE: You may be prompted to provide your password three times.

thingworxPostgresSchemaSetup.bat Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	localhost For RDS: rds-host	IP or host name of the database NOTE: For RDS: Hostname or IP of RDS PostgreSQL instance	-h localhost For RDS: -h rds-host
-p or -P	port	5432	Port number of PostgreSQL	-p 5432
-d or -D	database	thingworx	Database name to use	-d thingworx
-s or -S	schema	public	Schema name to use	-s mySchema
-u or -U	username	twadmin	Username to update the database schema	-u twadmin
-o or -O	option	all	There are three options: **all**: Sets up the model and data provider schemas into the specified database. **model**: Sets up the model provider schema into the specified database. **data**: Sets up the data provider schema into the specified database.	-o data

Configuring platform-settings.json (Windows)

1. To use the default ThingworxPlatform configuration directory, create a folder called **ThingworxPlatform** at the root of the drive where Tomcat was installed. Alternatively, if you want to specify the location where ThingWorx stores its settings, you can set the **THINGWORX_PLATFORM_SETTINGS** environment variable to the desired location.

   Ensure that the folder referenced by **THINGWORX_PLATFORM_SETTINGS** exists and is writable by the Tomcat user. This environment variable should be configured as part of the system environment variables.
2. Place the **platform-settings.json** file into the **ThingworxPlatform** folder. This file is available in the software download.

3. Open **platform-settings.json** and configure as necessary. Refer to the configuration options in Appendix C: platform-settings.json Options and Appendix B: platform-settings.json sample.

NOTE: If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and platform-settings.json file. The default installation assumes both servers are on the same machine.

Encrypting the PostgreSQL Password (Windows)

If you want to provide added security encryption for the PostgreSQL database settings in the platform-settings.json file, you can do the following.

Note: This encryption process is optional.

Prerequisites

- You must have Java installed and on your path.
- You must have PostgreSQL installed and know the password.

1. Create a working directory such as C:\password_setup (windows), and copy the Thingworx.war there.
2. Unzip the Thingworx.war.
3. Open a command prompt, cd to your working directory, and set your CLASSPATH by doing the following:
   a. Go to Control Panel > System Properties > Environment Variables.
   b. Create a new environment variable: PG_PW_UTIL
      `C:\password_setup\WEB-INF\lib\thingworx-platform-common-<release-version>.jar;C:\password_setup\WEB-INF\lib\slf4j-api-1.7.12.jar;C:\password_setup\WEB-INF\lib\logback-core-1.0.13.jar;C:\password_setup\WEB-INF\lib\logback-classic-1.0.13.jar;C:\password_setup\WEB-INF\lib\thingworx-common-<release-version>.jar`
   c. Add the variable to the CLASSPATH.
      `CLASSPATH`
      `<don't touch existing classpath>; %PG_PW_UTIL%`
   d. In your command shell, enter `'java -version'`.
      It should respond with a Java version.
4. Open /ThingworxPlatform/platform-settings.json and change the password value to `'encrypt.db.password'`.
   For example, `"password": "encrypt.db.password",`
5. Create a directory named `"\twx"` that has the same parent directory as the `"\ThingworxPlatform"` directory.
6. To create a key store with the PostgreSQL password encrypted inside, run the following command: `java com.thingworx.platform.security.keystore.ThingworxKeyStore encrypt.db.password <postgres_password>`
   The second argument must be your PostgreSQL password.
7. Once you have created the encrypted password, remove the updates to the CLASSPATH.

Installing ThingWorx (Windows)

1. Locate the appropriate **Thingworx.war** file.

NOTE: ThingWorx downloads are available in PTC Software Downloads.

2. Copy the **Thingworx.war** file and place it in the following location of your Tomcat installation:

   ```
   \Apache Software Foundation\Tomcat 8.0\webapps
   ```

3. To launch ThingWorx, go to **<servername>/Thingworx** in a web browser.
   NOTE: Use a strong password. The login information below is for the Administrator user only.

   Use the following login information:
   Login Name: Administrator
   Password: admin

Installing ThingWorx for the First Time: PostgreSQL or H2 on Ubuntu

Oracle Java, and Apache Tomcat, and PostgreSQL (if using PostgreSQL) must be installed prior to installing ThingWorx. Refer to the System Requirements and Compatibility Matrix for specific version requirements.

NOTE: This version of ThingWorx has been tested with Ubuntu 14.04. Other versions are not supported and may not work.

Installing Oracle Java and Apache Tomcat (Ubuntu)

1. Update Ubuntu packages:

   ```
   $ sudo apt-get update
   ```

2. Install and Configure Network Time Protocol (NTP) settings for time synchronization:

   ```
   $ sudo apt-get install ntp
   ```

   NOTE: The default configuration for NTP is sufficient. For additional configuration information about NTP (beyond the scope of this documentation), refer to the following resources:
   - Time Synchronization with NTP
   - How do I use pool.ntp.org?

3. Edit AUTHBIND properties to allow Tomcat to bind to ports below 1024:

```
$ sudo apt-get install authbind
```

4. Download the Java 8u92 JDK tar file from Oracle's website, or run the following:

```
$ wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-
securebackup-cookie" "http://download.oracle.com/otn-
pub/java/jdk/8u92-b14/jdk-8u92-linux-x64.tar.gz"
```

   NOTE**:** This version of ThingWorx has been tested with Java 8 update 92. Other versions are not supported and may not work*.*

5. Extract tar file:

```
$ tar -xf jdk-8u92-linux-x64.tar.gz
```

6. Create the directory by moving the JDK to **`/usr/lib/jvm`**:

```
$ sudo mkdir -p /usr/lib/jvm
$ sudo mv jdk1.8.0_92/ /usr/lib/jvm/
```

7. Add alternatives to the system:

```
$ sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/lib/jvm/jdk1.8.0_92/bin/java" 1
$ sudo update-alternatives --install "/usr/bin/keytool"
"keytool" "/usr/lib/jvm/jdk1.8.0_92/bin/keytool" 1
```

8. Change access permissions:

```
$ sudo chmod a+x /usr/bin/java
$ sudo chmod a+x /usr/bin/keytool
```

9. Change owner:

```
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_92/
```

10. Configure master links:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config keytool
```

    NOTE: "Nothing to configure" is a normal response to this command and is not an error.

    NOTE: Additional executables in /usr/lib/jvm/jdk1.8.0_92/bin/ can be installed using the previous set of steps.

11. Verify Java version:

```
$ java –version-
```

NOTE: This should return something similar to the following (build specifics may be different):

```
java version "1.8.0_92"
Java(TM) SE Runtime Environment (build 1.8.0_92-b14)
Java HotSpot(TM) 64-Bit Server VM (build 24.75-b04, mixed mode)
```

12. Download Apache Tomcat:

```
$ wget http://archive.apache.org/dist/tomcat/tomcat-
8/v8.0.33/bin/apache-tomcat-8.0.33.tar.gz
```

NOTE**:** This version of ThingWorx has been tested with Tomcat 8.0.33. Other versions are not supported and may not work.

13. Extract tar file:

```
$ tar -xf apache-tomcat-8.0.33.tar.gz
```

14. Move Tomcat to /usr/share/tomcat8 **:**

```
$ sudo mkdir -p /usr/share/tomcat8
$ sudo mv apache-tomcat-8.0.33 /usr/share/tomcat8/8.0.33
```

15. Define environment variables in /etc/environment:

```
$ export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_92
$ export CATALINA_HOME=/usr/share/tomcat8/8.0.33
```

16. Change directory to $CATALINA_HOME:

```
$ cd $CATALINA_HOME
```

17. Add user and group to the system:

```
$ sudo addgroup --system tomcat8 --quiet
$ sudo adduser --system --home /usr/share/tomcat8/ --no-create-
home --ingroup tomcat8 --disabled-password --shell /bin/false
tomcat8
```

18. Change owner and access permissions of bin/ lib/ and webapps/ :

```
$ sudo chown -Rh tomcat8:tomcat8 bin/ lib/ webapps/
$ sudo chmod 775 bin/ lib/ webapps/
```

19. Change owner and access permissions of conf/:

```
$ sudo chown -Rh root:tomcat8 conf/
$ sudo chmod 640 conf/*
```

20. Change access permissions of logs/, temp/, and work/:

```
$ sudo chown -R tomcat8:adm logs/ temp/ work/
$ sudo chmod 750 logs/ temp/ work/
```

21. In bin/, create setenv.sh with the following contents:

```
# Java Options
export JAVA_OPTS="-Djava.awt.headless=true -
Djava.net.preferIPv4Stack=true -Dserver -Dd64 -XX:+UseNUMA -
XX:+UseConcMarkSweepGC -Dfile.encoding=UTF-8"
```

NOTE: For more information on these options and for additional options for hosted and/or public-facing environments, refer to the Appendix: Tomcat Java Option Settings.

22. Change owner and access permissions of bin/setenv.sh:

```
$ sudo chown tomcat8:tomcat8 bin/setenv.sh
$ sudo chmod 775 bin/setenv.sh
```

23. Create self-signed certificate:

```
$ sudo $JAVA_HOME/bin/keytool -genkey -alias tomcat8 -keyalg RSA
-keystore $CATALINA_HOME/conf/.keystore
$ sudo chown root:tomcat8 $CATALINA_HOME/conf/.keystore
$ sudo chmod 640 $CATALINA_HOME/conf/.keystore
```

24. Uncomment the Manager element in $CATALINA_HOME/conf/context.xml to prevent sessions from persisting across restarts:

```
<Manager pathname="" />
```

25. Modify the shutdown string and protocol used by the SSL Connector in $CATALINA_HOME/conf/server.xml (comment out the non-SSL Connector):

```
<Server port="8005" shutdown="TH!nGW0rX">

<Connector port="443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
keystoreFile="${user.home}/8.0.33/conf/.keystore"
keystorePass="changeit" clientAuth="false" sslProtocol="TLS" />
```

26. Define a user in $CATALINA_HOME/conf/tomcat-users.xml:

    ```
    <user username="tomcat" password="tomcat" roles="manager"/>
    ```

    NOTE: In hosted and/or public-facing environments, use of the manager web application is not recommended because it introduces a security risk. Similarly, the example web applications included in /webapps should be removed as they may introduce unnecessary security vulnerabilities into Tomcat.

27. Determine uid of tomcat8 user:

    ```
    $ id -u tomcat8
    110
    ```

28. Using this number, create an ID file In /etc/authbind/byuid/

    NOTE: The number returned may be different than this example. These examples use 110. Change this to what was returned in the previous step.

    ```
    $ sudo touch /etc/authbind/byuid/110
    ```

    Edit this file and paste in the following:

    ```
    0.0.0.0/0:1,1023
    ```

29. Change owner and access permissions of /etc/authbind/byuid/110:

    ```
    $ sudo chown tomcat8:tomcat8 /etc/authbind/byuid/110
    $ sudo chmod 700 /etc/authbind/byuid/110
    ```

30. Modify $CATALINA_HOME/bin/startup.sh to always use authbind:

    ```
    #exec "$PRGDIR"/"$EXECUTABLE" start "$@"
    exec authbind --deep "$PRGDIR"/"$EXECUTABLE" start "$@"
    ```

31. In /etc/init.d, create tomcat8 file:

```
$ sudo touch /etc/init.d/tomcat8
```

Edit the file and enter the following contents:

```
CATALINA_HOME=/usr/share/tomcat8/8.0.33

case $1 in
  start)
    /bin/su -p -s /bin/sh tomcat8 $CATALINA_HOME/bin/startup.sh
  ;;

  stop)
    /bin/su -p -s /bin/sh tomcat8 $CATALINA_HOME/bin/shutdown.sh
  ;;

  restart)
    /bin/su -p -s /bin/sh tomcat8 $CATALINA_HOME/bin/shutdown.sh
    /bin/su -p -s /bin/sh tomcat8 $CATALINA_HOME/bin/startup.sh
  ;;

esac
exit 0
```

32. Change access permissions of etc/init.d/tomcat8 and create symbolic links:

```
$ sudo chmod 755 /etc/init.d/tomcat8
$ sudo ln -s /etc/init.d/tomcat8 /etc/rc1.d/K99tomcat
$ sudo ln -s /etc/init.d/tomcat8 /etc/rc2.d/S99tomcat
```

OPTIONAL STEP: If you want to increase the default cache settings that affect static file caching, add the following line within the <context></context> tags in the $TOMCAT_HOME/conf/context.xml file:

```
<Resources cacheMaxSize="501200" cacheObjectMaxSize="2048"
cacheTtl="60000"/>
```

33. If you are installing H2, skip to the Installing ThingWorx section.

Installing and Configuring PostgreSQL (Ubuntu)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers).

NOTE: If you are including the HA layer to your implementation, refer to the ThingWorx High Availability Administrator's Guide.

Installing PostgreSQL and Creating a New User Role in PostgreSQL (Ubuntu)

1. Download and install the appropriate version of PostgreSQL.

   In Ubuntu 14.4, the PostgreSQL repository can be added allowing the application to be installed directly from the package manager::

   ```
   $ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/
   trusty-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

   $ sudo wget -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
   sudo apt-key add -

   $ sudo apt-get update

   $ sudo apt-get install postgresql-9.4 -y
   ```

   NOTE: This version of ThingWorx has been tested with PostgreSQL versions 9.4.5 through 9.4.10. Other versions are not supported and may not work.

2. Install PgAdmin III, the PostgreSQL admin tool:

   ```
   $ sudo apt-get install pgadmin3 -y
   ```

NOTE: PgAdmin III is an open source management tool for your databases that is included in the PostgreSQL download. The tool features full Unicode support, fast, multithreaded query, and data editing tools and support for all PostgreSQL object types.

3. Set up password for the postgres user:

   ```
   $ sudo passwd postgres
   ```

4. Enter the password for the postgres user. Take note of this password.

   NOTE: The password is *password* in the following example.

5. Set up postgres user in psql:

   ```
   $ sudo -u postgres psql -c "ALTER ROLE postgres WITH password
   'password'"
   ```

   NOTE: The password should be the same as in the step above.

6. Configure pgadmin3:

   ```
   $ sudo pgadmin3
   ```

   * In the pgAdminIII GUI, click on file->Open postgresql.conf
   * Open /etc/postgresql/9.4/main/postgresql.conf
   * Put a check next to listen addresses and port
   -The default settings of "localhost" and "5432" are usually sufficient.
   * Save and close.
   * Click on file->Open pg_hba.conf
   * Open /etc/postgresql/9.4/main/pg_hba.conf
   * Double-click on the database ‘all’ line with address 127.0.0.1/32
   * Set Method to md5
   * Double-click on the database ‘all’ line with address 1/128
   * Set Method to md5
   * Click OK
   * Save and exit
   * Close pgadmin3

7. Restart the PostgreSQL service:

   ```
   $ sudo service postgresql restart
   ```

8. Set up Pgadmin III to connect to the database:

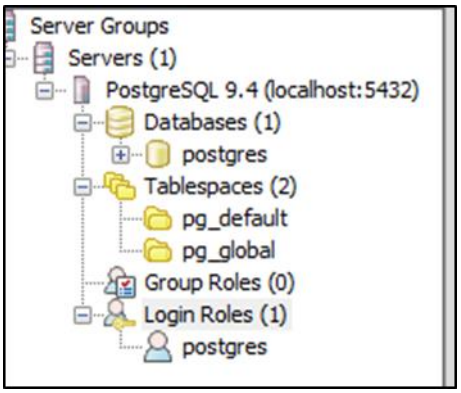
   ```
   $ sudo pgadmin3
   ```

9. Click the plug Add a connection to a server in the top left corner:

   Fill out the following:

   ```
   Name: PostgreSQL 9.4
   Host: localhost
   Port: 5432
   Service: <blank>
   Maintenance DB: postgres
   Username: postgres
   Password: <password as set in step above>
   Store password: Checked
   Group: Servers
   ```

10. Click **OK**.
11. Create a new user role (in this example, it is twadmin):
    a. Right click **PostgreSQL9.4**(localhost:5432).
    b. Select **NewObject>NewLogin Role**. On the **Properties** tab, in the **Role name** field, type twadmin.
    c. On the **Definition** tab, in the **Password** field, type password (must type twice).
12. Click **OK**.
    NOTE: Remember the user role name and password created in this step for later use.



Configuring and Executing the PostgreSQL Database Script (Ubuntu)

To set up the PostgreSQL database and tablespace, the **thingworxPostgresDBSetup.sh** script must be configured and executed.

1. Create a directory named **ThingworxPostgresqlStorage** on your Thingworx Storage drive.

   ```
   $ sudo mkdir /ThingworxPostgresqlStorage
   $ sudo chown postgres:postgres /ThingworxPostgresqlStorage
   $ sudo chmod 755 /ThingworxPostgresqlStorage
   ```

   NOTE: You must specify the **-l** option to a path that exists. For example, **-l ThingworxPostgresqlStorage**. The script does not create the folder for you.
   As shown above, this folder needs have appropriate ownership and access rights. It should be owned by the postgres user and have the read, write, and execute assigned to the owner.

NOTE: If you create with the -**d<databasename>**, you do not have to use the postgres user.

2. Obtain and open **thingworxPostgresDBSetup.sh** from the ThingWorx software download.
3. Configure the script. Reference the configuration options in the table below.

   NOTE: This example uses the 7.2.0 download from the PTC site. If necessary, change the file name to the version you are using.

```
$ sudo unzip MED-61111-CD-072_F000_ThingWorx-Platform-Postgres-7-2-
0.zip
$ cd install
```

   Various parameters such as **server**, **port**, **database**, **tablespace**, **tablespace location** and **thingworxusername** can be configured in the script, depending on the requirements. Execute this script with the **--help** option for usage information.

   As an example, to set up the database and tablespace with a default Postgres installation that has a postgres database as well as a postgres user name and assuming the user created above is **twadmin**, enter:

```
$ sudo sh thingworxPostgresDBSetup.sh -a postgres -u twadmin -l
/ThingworxPostgresqlStorage
```

thingworxPostgresDBSetup.sh Script Options

Option	Parameter	Default	Description	Example
-t or -T	server	localhost	Tablespace name	-t thingworx
-p or -P	port	5432	Port number of PostgreSQL	-p 5432
-d or -D	database	thingworx	PostgreSQL Database name to create	-d thingworx
-h or -H	tablespace	thingworx	Name of the PostgreSQL tablespace.	-h localhost
-l or -L	tablespace_location	/ThingworxPostgresqlStorage	Required. Location in the file system where the files representing database objects are stored. *	-l or -L
-a or -A	adminusername	postgres	Administrator Name	-a postgres

Option	Parameter	Default	Description	Example
-u or -U	thingworxusername	twadmin	User name that has permissions to write to the database.	-u twadmin

Configuring and Executing the Model/Data Provider Schema Script (Ubuntu)

To set up the PostgreSQL model/data provider schema, the **thingworxPostgresSchemaSetup.sh** script must be configured and executed. This will set up the public schema under your database on the PostgreSQL instance installed on the localhost.

1. Obtain and open the **thingworxPostgresSchemaSetup.sh** from the ThingWorx software download package.
2. Configure the script. Reference the configuration options in the table below or execute the script with **--help** option for usage information. The script can be run with the default parameters as:

```
$ sudo sh thingworxPostgresSchemaSetup.sh
```

thingworxPostgresSchemaSetup.sh Script Options

Option	Parameter	Default	Description	Example
-h or -H	server	localhost For RDS: rds-host	IP or host name of the database For RDS: Hostname or IP of RDS PostgreSQL instance	-h localhost For RDS: -h rds-host
-p or -P	port	5432	Port number of PostgreSQL	-p 5432
-d or -D	database	thingworx	Database name to use	-d thingworx
-s or -S	schema	public	Schema name to use	-s mySchema
-u or -U	username	twadmin	Username to update the database schema	-u twadmin
-o or -O	option	all	There are three options: **all**: Sets up the model and data provider schemas into the specified database. **model**: Sets up the model provider	-o data

Option	Parameter	Default	Description	Example
			schema into the specified database. **data**: Sets up the data provider schema into the specified database.	

Configuring platform-settings.json (Ubuntu)

1. Create a folder called **ThingworxPlatform** at the root (for example, **/ThingworxPlatform** or as a system variable (for example, **THINGWORX_PLATFORM_SETTINGS=/data/ThingworxPlatform**).

   ```
   $ sudo mkdir /ThingworxPlatform
   ```

2. Copy the **platform-settings.json file from the install package to the /ThingworxPlatform** folder. From the directory containing the JSON file:

   ```
   $ sudo cp platform-settings.json /ThingworxPlatform/
   ```

   The JSON file contains default settings. Refer to alternate configuration options in: Appendix C: platform-settings.json Options and Appendix B: platform-settings.json sample.

   NOTE: If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and platform-settings.json file. The default installation assumes both servers are on the same machine.

(OPTIONAL) Encrypting the PostgreSQL Password (Ubuntu)

If you want to provide added security encryption for the PostgreSQL database settings in the platform-settings.json file, you can do the following.
Note: This encryption process is optional.

Prerequisites

- You must have Java installed and on your path.
- You must have PostgreSQL installed and know the password.

1. Create a working directory such as ~/password_setup, and copy the Thingworx.war there.

2. Unzip the Thingworx.war.

3. Open a command prompt, cd to your working directory, and set your CLASSPATH:

   ```
   $ export CLASSPATH=WEB-INF/lib/thingworx-platform-common-<release-
   version>.jar:WEB-INF/lib/slf4j-api-1.7.12.jar:WEB-INF/lib/logback-core-
   ```

```
1.0.13.jar:WEB-INF/lib/logback-classic-1.0.13.jar:./WEB-
INF/lib/thingworx-common-<release-version>.jar
```

4. Open /ThingworxPlatform/platform-settings.json and change the password value to `'encrypt.db.password'`.

   For example:

   ```
   "password": "encrypt.db.password",
   ```

5. Create a directory named `"/twx"` that has the same parent directory as `"/ThingworxPlatform"`:

   ```
   $ sudo mkdir /twx
   ```

6. To create a key store with the PostgreSQL password encrypted inside, run the following command:

   ```
   $ java com.thingworx.platform.security.keystore.ThingworxKeyStore
   encrypt.db.password <postgres_password>
   ```

   NOTE: The second argument must be your PostgreSQL password.

7. Run the following commands:

   ```
   $ sudo chown tomcat8:tomcat8 /twx
   $ sudo chmod 755 /twx
   ```

Installing ThingWorx (Ubuntu)

1. Create **/ThingworxStorage** and **/ThingworxBackupStorage** directories:

   ```
   $ sudo mkdir /ThingworxStorage /ThingworxBackupStorage
   ```

2. Change owner and access permissions of **/ThingworxStorage** and **/ThingworxBackupStorage**:

   ```
   $ sudo chown tomcat8:tomcat8 /ThingworxStorage
   /ThingworxBackupStorage
   $ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage
   ```

3. Move the **thingworx.war** to **$CATALINA_HOME/webapps:**

   ```
   $ sudo mv Thingworx.war $CATALINA_HOME/webapps
   $ sudo chown tomcat8:tomcat8 $CATALINA_HOME/webapps/Thingworx.war
   $ sudo chmod 775 $CATALINA_HOME/webapps/Thingworx.war
   ```

4. Start Tomcat to deploy the ThingWorx web application:

```
$ sudo service tomcat8 start
```

5. You should now be able to connect to ThingWorx Composer through a web browser at **https://localhost/Thingworx**

   Username: **Administrator**
   Password: **admin**

   NOTE**:** It is normal when using a self-signed certificate to get a security warning when accessing ThingWorx. In most browsers, clicking on the advanced options will allow you to set an exception for your private certificate. Setting up a security certificate through a signing authority will allow secure connections without having to set an exception.

Installing and Configuring ThingWorx for the First Time: PostgreSQL or H2 on Red Hat Enterprise Linux (RHEL)

Oracle Java, Apache Tomcat, and PostgreSQL (if using PostgreSQL) must be installed prior to installing ThingWorx.

NOTE: This version of ThingWorx has been tested on RHEL 7.1. Other RHEL versions are not supported and may not work.

Installing Oracle Java and Apache Tomcat (RHEL)

1. Download the Java 8u92 (JDK) RPM file from Oracle's website, or run the following:

```
$ wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-
securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u92-
b14/jdk-8u92-linux-x64.rpm
```

2. Run the Java installer:

```
$ sudo rpm -i jdk-8u92-linux-x64.rpm
```

3. Create the directory and move the JDK:

```
$ sudo mkdir -p /usr/lib/jvm
$ sudo mv /usr/java/jdk1.8.0_92/ /usr/lib/jvm/
```

4. Set the Java alternatives:

```
$ sudo alternatives --install /usr/bin/java java
/usr/lib/jvm/jdk1.8.0_92/bin/java 1
$ sudo alternatives --install /usr/bin/keytool keytool
/usr/lib/jvm/jdk1.8.0_92/bin/keytool 1
```

5. Change access permissions:

```
$ sudo chmod a+x /usr/bin/java
$ sudo chmod a+x /usr/bin/keytool
```

6. Change Owner:

```
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0_92/
```

7. Configure master links:

```
$ sudo alternatives --config java
```

NOTE: Select the option that contains /usr/lib/jvm/jdk1.8.0_92/bin/java

```
$ sudo rm /usr/java/latest
$ sudo ln -s /usr/lib/jvm/jdk1.8.0_92 /usr/java/latest
$ sudo ln -s /usr/lib/jvm/jdk1.8.0_92/bin/keytool /usr/bin/keytool
```

NOTE: *This may return a 'File Exists' error. If so, ignore and continue.*

```
$ sudo alternatives --config keytool
```

8. Verify Java version:

```
$ java -version
java version "1.8.0_92"
Java(TM) SE Runtime Environment (build 1.8.0_92-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
```

9. Install Tomcat. Download the Tomcat installer:

```
$ wget https://archive.apache.org/dist/tomcat/tomcat-
8/v8.0.33/bin/apache-tomcat-8.0.33.tar.gz
```

10. Extract the contents:

```
$ tar -xf apache-tomcat-8.0.33.tar.gz
```

11. Move Tomcat to **/usr/share/tomcat8**:

```
$ sudo mkdir -p /usr/share/tomcat8
$ sudo mv apache-tomcat-8.0.33 /usr/share/tomcat8/8.0.33
```

12. Change directory to **/usr/share/tomcat8/8.0.33**:

```
$ cd /usr/share/tomcat8/8.0.33
```

13. Add user and group to the system:

```
$ sudo groupadd -r tomcat8
$ sudo useradd -r -d /usr/share/tomcat8 -g tomcat8 -s /bin/false
tomcat8
```

14. Change owner and access permissions of **bin/ lib/** and **webapps/:**

```
$ sudo chown -Rh tomcat8:tomcat8 bin/ lib/ webapps/
$ sudo chmod 775 bin/ lib/ webapps/
```

15. Change owner and access permissions of **conf/**:

```
$ sudo chown -Rh root:tomcat8 conf/
$ sudo chmod 640 conf/*
```

16. Change access permissions of **logs/**, **temp/**, and **work/**:

```
$ sudo chown -R tomcat8:adm logs/ temp/ work/
$ sudo chmod 750 logs/ temp/ work/
```

17. Create a **bin/setenv.sh** file:

```
$ sudo touch bin/setenv.sh
```

Open **bin/setenv.sh** in an editor (as root), paste the following and save:

```
# Java Options
export JAVA_OPTS="-Djava.awt.headless=true -
Djava.net.preferIPv4Stack=true -Dserver -Dd64 -XX:+UseNUMA -
XX:+UseConcMarkSweepGC -Dfile.encoding=UTF-8" export
JRE_HOME=/usr/lib/jvm/jdk1.8.0_92/jre
```

NOTE: For more information on these options and for additional options for hosted and/or public-facing environments, refer to the Appendix: Tomcat Java Option Settings.

18. Change owner and access permissions of **bin/setenv.sh**:

```
$ sudo chown tomcat8:tomcat8 bin/setenv.sh
$ sudo chmod 775 bin/setenv.sh
```

19. Create self-signed certificate:

```
$ /usr/lib/jvm/jdk1.8.0_92/jre/bin/keytool -genkey -alias tomcat8 -
keyalg RSA
```

Follow the instructions to complete the certificate creation process.
Set the keystore password to **changeit**
Follow the prompts to set up your security certificate.
Set the tomcat8 user password to the same as the keystore password

```
$ sudo cp ~/.keystore /usr/share/tomcat8/8.0.33/conf/
$ sudo chown root:tomcat8 /usr/share/tomcat8/8.0.33/conf/.keystore
$ sudo chmod 640 /usr/share/tomcat8/8.0.33/conf/.keystore
```

20. Uncomment the Manager element in **context.xml** to prevent sessions from persisting across restarts.

Open **/usr/share/tomcat8/8.0.33/conf/context.xml** in a text editor (as root) and remove the '<!—' before '<Manager pathname="" />' and the '-->' after

21. Save the file.

22. Modify the shutdown string and protocol used by the SSL Connector in **server.xml**:

Open **/usr/share/tomcat8/8.0.33/conf/server.xml** in a text editor (as root)

Change '<Server port="8005" shutdown="SHUTDOWN">' to '<Server port="8005" shutdown="TH!nGW0rX ">'

Comment out or remove this section:

```
' <Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000" redirectPort="8443" />'
```

Paste in this section directly below:

```
<Connector port="443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true" scheme="https"
secure="true"
   keystoreFile="${user.home}/8.0.33/conf/.keystore"
   keystorePass="changeit" clientAuth="false"
   sslProtocol="TLS" />
```

23. Define an Apache Manager user in tomcat-users.xml:

    Open **/usr/share/tomcat8/8.0.33/conf/tomcat-users.xml** in a text editor (as root)
    Just above the final line (</tomcat-users>) add the following line:

    ```
    <user username="tomcat" password="tomcat"
    roles="manager,manager-gui"/>
    ```

24. Save the file.
    NOTE: The roles included are for ease of testing and can be removed if security is a concern.

25. Set up Tomcat as a service to start on boot. First, build JSVC:

    ```
    $ sudo yum install gcc
    ```

    NOTE: This may already be installed on your system. If so, continue.

    ```
    $ cd /usr/share/tomcat8/8.0.33/bin/
    $ sudo tar xvfz commons-daemon-native.tar.gz
    $ cd commons-daemon-*-native-src/unix
    $ sudo ./configure --with-java=/usr/java/latest
    $ sudo make
    $ sudo cp jsvc ../..
    ```

26. Create the Tomcat service file:

```
$ sudo touch /usr/lib/systemd/system/tomcat.service
```

Open **/usr/lib/systemd/system/tomcat.service** in a text editor (as root) and paste in the following:

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking
PIDFile=/var/run/tomcat.pid
Environment=CATALINA_PID=/var/run/tomcat.pid
Environment=JAVA_HOME=/usr/lib/jvm/jdk1.8.0_92
Environment=CATALINA_HOME=/usr/share/tomcat8/8.0.33
Environment=CATALINA_BASE=/usr/share/tomcat8/8.0.33
Environment=CATALINA_OPTS=

ExecStart=/usr/share/tomcat8/8.0.33/bin/jsvc \
                  -Dcatalina.home=${CATALINA_HOME} \
                  -Dcatalina.base=${CATALINA_BASE} \
                  -cp ${CATALINA_HOME}/bin/commons-
daemon.jar:${CATALINA_HOME}/bin/bootstrap.jar:${CATALINA_HOME}/bin/tomcat
-juli.jar \
                  -user tomcat8 \
                  -java-home ${JAVA_HOME} \
                  -pidfile /var/run/tomcat.pid \
                  -errfile ${CATALINA_HOME}/logs/catalina.out \
                  -outfile ${CATALINA_HOME}/logs/catalina.out \
                  $CATALINA_OPTS \
                  org.apache.catalina.startup.Bootstrap

ExecStop=/usr/share/tomcat8/8.0.33/bin/jsvc \
                  -pidfile /var/run/tomcat.pid \
                  -stop \
                  org.apache.catalina.startup.Bootstrap

[Install]
WantedBy=multi-user.target
```

27. Set Tomcat to run on system start up:

```
$ sudo systemctl enable tomcat.service
```

Note: This will allow the user to control the Tomcat service with the following commands:

systemctl start tomcat
systemctl stop tomcat
systemctl restart tomcat
systemctl status tomcat

28. Start the Tomcat service and test:

```
$ sudo systemctl start tomcat
```

You should now be able to connect to the Tomcat server by entering **https://localhost** in a browser.

29. If you are installing H2, skip to the Installing ThingWorx section.

Installing and Configuring PostgreSQL (RHEL)

The instructions provided below are intended for the PostgreSQL administrator (not the DB host servers).

NOTE: This guide assumes a version of RHEL with a GUI (X11) and an active account with access to the RHEL software repositories.

If you are working without a GUI, skip installing PgAdmin III and refer to this support article for alternate instructions. If you do not have access to the official RHEL software sources, you can set up a free open source repository from the EPEL team. (this site is not provided or controlled by PTC).

NOTE: If you are including the HA layer to your implementation, refer to the ThingWorx High Availability Administrator's Guide.

Installing PostgreSQL and Creating a New User Role in PostgreSQL (RHEL)

1. Add the PostgreSQL repository to Yum and install:

   ```
   $ sudo rpm -Uvh
   http://download.postgresql.org/pub/repos/yum/9.4/redhat/rhel-7-
   x86_64/pgdg-redhat94-9.4-3.noarch.rpm
   $ sudo yum install postgresql94 postgresql94-server postgresql94-
   contrib
   ```

   NOTE: This version of ThingWorx has been tested with PostgreSQL 9.4. Other versions are not supported and may not work.
2. Install PgAdmin III:

   ```
   $ sudo yum install pgadmin3
   ```

3. Initialize and launch the database:

   ```
   $ sudo /usr/pgsql-9.4/bin/postgresql94-setup initdb
   ```

4. Set the PostgreSQL service to start on boot:

   ```
   $ sudo chkconfig postgresql-9.4 on
   $ sudo service postgresql-9.4 start
   ```

5. Set up password for the postgres user:

   ```
   $ sudo passwd postgres
   ```

   Enter the password for the postgres user
   Take note of this password.

6. Set up postgres user in psql:

   ```
   $ sudo -u postgres psql -c "ALTER ROLE postgres WITH password
   'password'"
   ```

   NOTE: The password should be the same as in the step above.

7. Configure postgresql.conf with pgadmin3.

   ```
   $ sudo pgadmin3
   ```

   In the pgAdminIII GUI, click on **file->Open postgresql.conf**

   * Open **/var/lib/pgsql/9.4/data/postgresql.conf**
   * Put a check next to **listen addresses** and **port**
     -The default settings of **localhost** and **5432** are usually sufficient.

   Save and close

9. Configure **pg_hba.conf** with pgadmin3

   * Click on **file->Open pg_hba.conf**
   * Open **/var/lib/pgsql/9.4/data/pg_hba.conf**
   * Double-click on the line with address **127.0.0.1/32**
   * Set Method to **md5**
   * Double-click on the line with address **1/128**
   * Set Method to **md5**
   * Click **OK**
   * Save and exit
   * Close pgadmin3

8. Restart the PostgreSQL service:

   ```
   $ sudo service postgresql-9.4 restart
   ```

9. Set up pgadmin3 to connect to the database.

   ```
   $ sudo pgadmin3
   ```

   Click the plug **Add a connection to a server** in the top left corner.
   Fill out the following:

   ```
   Name: PostgreSQL 9.4
   Host: localhost
   Port: 5432
   Service: <blank>
   Maintenance DB: postgres
   Username: postgres
   Password: <password as set in step above>
   Store password: Checked
   Group: Servers
   ```

   Click **OK**

10. Create a new user role (in this example, it is **twadmin**):

    Right click **PostgreSQL9.4 (localhost:5432**).
    Note: It may be possible to activate some extensions. Click **Databases** and select **postgres** in the main window. A dialog displays. Click **Fix it!**
    Select **NewObject>New Login Role**.
    On the **Properties** tab, in the **Role name** field, type **twadmin**.
    On the **Definition** tab, in the **Password** field, type password (must repeat in the Password (again) field).
    Click **OK**.
    Close pgAdmin3

11. Click **OK**.

12. Create the **ThingworxPostgresqlStorage** directory:

```
$ sudo mkdir /ThingworxPostgresqlStorage
$ sudo chmod 775 /ThingworxPostgresqlStorage
$ sudo chown postgres:postgres /ThingworxPostgresqlStorage/
$ sudo mkdir /ThingworxPlatform
$ sudo chmod 775 /ThingworxPlatform
$ sudo chown tomcat8:tomcat8 /ThingworxPlatform
```

13. Download the ThingWorx installer from the PTC downloads page:

```
https://support.ptc.com/appserver/auth/it/esd/index.jsp
```

    Unzip and open a terminal in the */installer/install directory.

14. Execute the PostgreSQL Database Script:

```
$ sudo sh thingworxPostgresDBSetup.sh -a postgres -u twadmin -l
/ThingworxPostgresqlStorage
```

15. Execute the Model/Data Provider Schema Script:

```
$ sh thingworxPostgresSchemaSetup.sh
```

    NOTE: When prompted, use the password for twadmin that was previously set up.

16. Startup configuration of platform-settings.json:

```
$ sudo cp ../platform-settings.json /ThingworxPlatform/
```

NOTE: If your PostgreSQL server is not the same as your ThingWorx server, and you are having issues with your ThingWorx installation, review your Tomcat logs and platform-settings.json file. The default installation assumes both servers are on the same machine.

(OPTIONAL) Encrypting the PostgreSQL Password (RHEL)

If you want to provide added security encryption for the PostgreSQL database settings in the platform-settings.json file, you can do the following.

Prerequisites

- You must have Java installed and on your path.
- You must have PostgreSQL installed and know the password.

1. Create a working directory such as ~/password_setup, and copy the Thingworx.war there.
2. Unzip the Thingworx.war.
3. Open a command prompt, cd to your working directory, and set your CLASSPATH to the following:
   $ `export CLASSPATH=WEB-INF/lib/thingworx-platform-common-<release-version>.jar:WEB-INF/lib/slf4j-api-1.7.12.jar:WEB-INF/lib/logback-core-1.0.13.jar:WEB-INF/lib/logback-classic-1.0.13.jar:./WEB-INF/lib/thingworx-common-<release-version>.jar`
4. Open /ThingworxPlatform/platform-settings.json and change the password value to `'encrypt.db.password'`.
   For example, `"password": "encrypt.db.password",`
5. Create a directory named `"/twx"` that has the same parent directory as `"/ThingworxPlatform"`:
   $ `sudo mkdir /twx`
6. To create a key store with the PostgreSQL password encrypted inside, run the following command:
   $ `java com.thingworx.platform.security.keystore.ThingworxKeyStore encrypt.db.password <postgres_password>`

   NOTE: The second argument must be your PostgreSQL password.
7. Run the following commands:
   $ `sudo chown tomcat8:tomcat8 /twx`
   $ `sudo chmod 755 /twx`

Installing ThingWorx (RHEL)

1. Create `/ThingworxStorage` and `/ThingworxBackupStorage` directories:

```
$ sudo mkdir /ThingworxStorage /ThingworxBackupStorage
```

2. Change owner and access permissions of `/ThingworxStorage` and `/ThingworxBackupStorage`:

```
$ sudo chown tomcat8:tomcat8
/ThingworxStorage/ThingworxBackupStorage
$ sudo chmod 775 /ThingworxStorage /ThingworxBackupStorage
```

3. Move **Thingworx.war** to the Tomcat /webapps directory:

   Change your working directory to the **installer** folder in the unzipped ThingWorx download package and run these commands:

```
$ sudo mv Thingworx.war /usr/share/tomcat8/8.0.33/webapps/
$ sudo chown tomcat8:tomcat8
/usr/share/tomcat8/8.0.33/webapps/Thingworx.war
$ sudo chmod 775 /usr/share/tomcat8/8.0.33/webapps/Thingworx.war
```

4. Restart Tomcat to start ThingWorx:

```
$ sudo systemctl restart tomcat
```

5. Log into ThingWorx Composer:

   In a browser, open **https://localhost/Thingworx**

   NOTE: Use a strong password. The login information below is for the Administrator user only.
   User: **Administrator**
   Password: **admin**

PostgreSQL Installation and Configuration with Amazon RDS

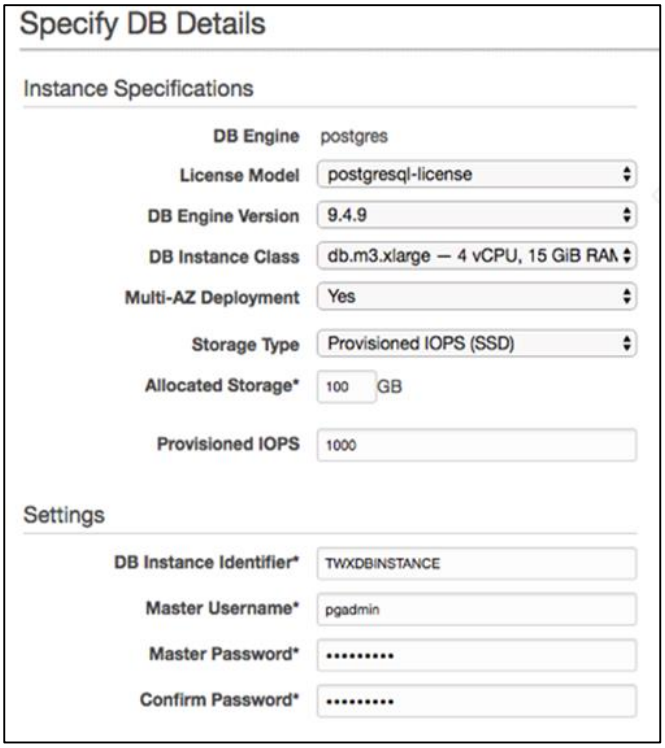
Installing PostgreSQL RDS Instance

NOTE: Before performing this section, perform the steps for installing Apache Tomcat and Java for your setup:

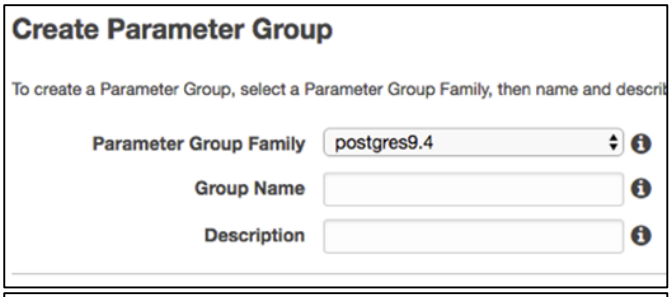
- Windows: Installing Oracle Java and Apache Tomcat
- Ubuntu: Installing Oracle Java and Apache Tomcat
- RHEL: Installing Oracle Java and Apache Tomcat

1. Follow the steps outlined in the Amazon RDS installation guide. The steps below provide supplemental guidance when you are ready to configure the **Specify DB Details** page in AWS.

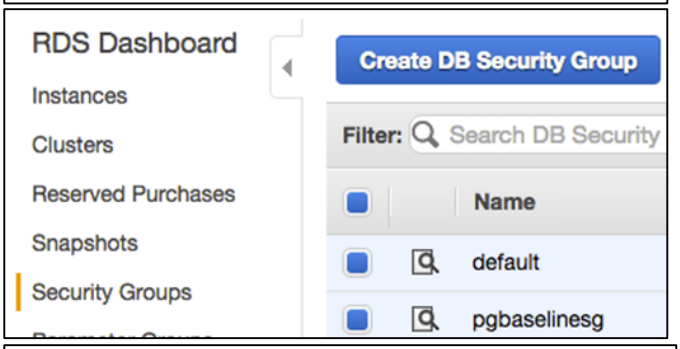
2. On the **Specify DB Details** page, specify the following information:
    a. In the **DB Engine Version** field, select the latest 9.4 version available. (9.4.9 in this example).
    b. In the **DB Instance Class** field, select the appropriate class. NOTE: **m3.2xlarge** is recommended for production use.
    c. In the **Mutli-AZ Deployme**nt field, select **Yes** if you have an HA environment.
    d. Note the **DB Instance Identifier** and **Master Username** for later use.

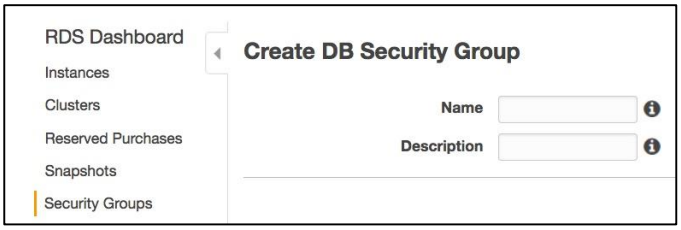


3. On the RDS Dashboard, click **Parameter Groups>Create DB Parameter Group**.
4. In the **Parameter Group Family** field, create a **Group Name** and **Description** for PostgreSQL database configuration later.



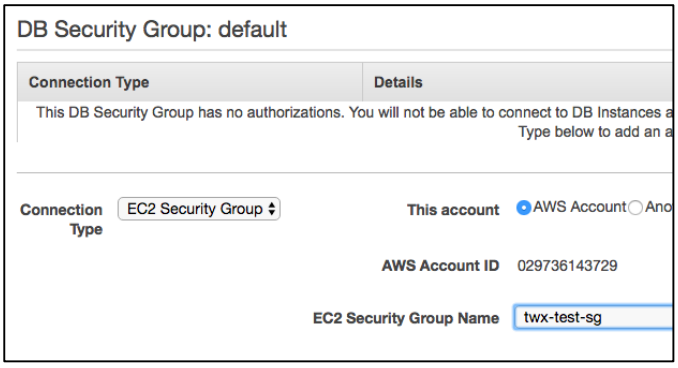
5. On the RDS Dashboard, click **Security Groups**.
6. Create a DB security group to control the DB access later.



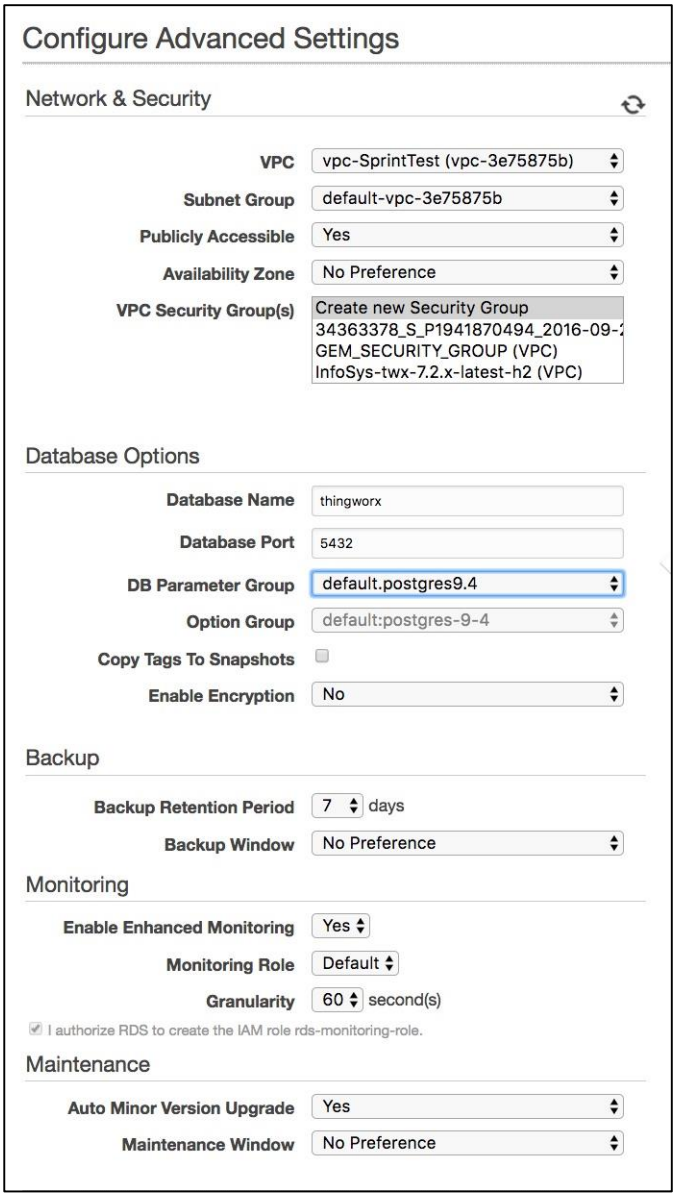


7. In the default **DB Security Group**, select the security group that the ThingWorx server will be using to allow access from the ThingWorx server to the database server.

   NOTE: This is not the same security group that was created in the previous step. This security group must be created in the EC2 section of AWS with the appropriate inbound/outbound rules to allow the postgres port to connect to the security group. This security group should also be assigned to the ThingWorx server instance.



8. On the **Configure Advanced Settings** page, specify the following information:
    a. In the **Network & Security** section, the settings should reflect the overall security configuration of the ThingWorx deployment environment (not specific to the database).
    NOTE: The **VPC** and **VPC Security Group(s)** should be created prior to installing the RDS database.
    b. In the **Database Options** section, type **thingworx** as the **Database Name**.
    NOTE: **thingworx** is the default name that is used in the schema creation scripts.
    c. In the **DB Parameter Group** field, select the name of the parameter group created previously.
    d. In the **Enable Encryption** field, select **Yes** if necessary.
    e. In the **Backup**, **Monitoring**, and **Maintenance** sections, select the appropraite options per your organizational needs.



Creating a New User Role in PostgreSQL (RDS)

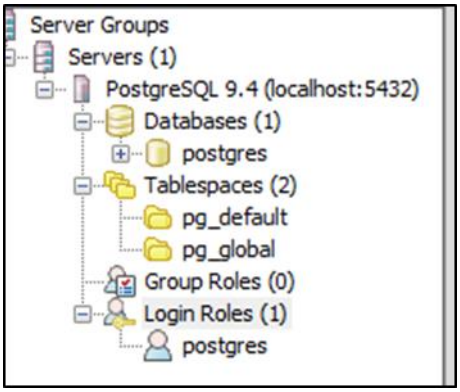
A PgAdmin lll client is required to connect to the Amazon RDS PostgreSQL instance and configure it. Install PgAdmin per the client machine's operating system before creating a new user role in RDS:

- Installing PgAdmin III (Windows)
- Installing PgAdmin III (Ubuntu)
- Installing PgAdmin III (RHEL)

1. Create a new user role. (in this example, it is **twadmin**):

   NOTE: The admin user created above while installing the RDS instance is **pgadmin**.

   a. Right click **PostgreSQL9.4**.
   b. Select **NewObject>New Login Role**. On the **Properties** tab, in the **Role name** field, type **twadmin**.
   c. On the **Definition** tab, in the **Password** field, type **password** (must type twice).
   a. Click **OK**.
   NOTE: Remember the user role name created (i.e. **twadmin**) in this step for later use.



Configuring and Executing the RDS PostgreSQL Database Script (Windows)

To set up the PostgreSQL database, the **thingworxRdsPostgresDBSetup.bat** script must be configured and executed.

NOTE: If you are using Ubuntu or RHEL, skip this section and go to:

- Ubuntu: Configuring and Executing the PostgreSQL Database Script
- RHEL: Configuring and Executing the PostgreSQL Database Script

1. Add the **<pgadmin-installation>/bin** folder to your system path variable.

2. Obtain and open **thingworxRdsPostgresDBSetup.bat** from the ThingWorx software download package.
3. Configure the script. Reference the configuration options in the table below.

   Various parameters such as **server**, **port**, **admindatabase, databaseadminusername, database** and **thingworxusername** can be configured in the script, depending on the requirements. Execute this script with the **--help** option for usage information.

   As an example, to set up the database and a postgres user name, assuming the user created above is **twadmin**, enter:

   ```
   thingworxRdsPostgresDBSetup –a pgadmin –u twadmin–b postgres –h rds-host
   ```

   where **twadmin** is the user name and **pgadmin** is the administrator account created during RDS installation

4. Execute the script. Once executed, this creates a new database in the Amazon RDS PostgreSQL instance.

   NOTE: You may need to run the command prompt as admin.

thingworxRdsPostgresDBSetup.bat Script Options

Option	Parameter	Default	Description	Example
-h or -H	Server name or IP	localhost	Hostname or IP of RDS PostgreSQL instance	-h rds-host
-p or -P	port	5432	Port number of PostgreSQL	-p 5432
-d or -D	database	thingworx	PostgreSQL Database name to create	-d thingworx
-b or -B	admindatabase	postgres	Name of the PostgreSQL system database	-b postgres
-a or -A	adminusername	postgres	Administrator Name	-a pgadmin
-u or -U	thingworxusername	twadmin	User name that has permissions to write to the database.	-u twadmin

Configuring and Executing the PostgreSQL Database Script (Ubuntu/RHEL)

To set up the PostgreSQL database and tablespace, the **thingworxRdsPostgresDBSetup.sh** script must be configured and executed.

1. Add the **<pgadmin-installation>/bin** folder to your system path variable.

2. Obtain and open **thingworxRdsPostgresDBSetup.sh** from the ThingWorx software download package.
3. Configure the script. Reference the configuration options in the table below.

   Various parameters such as **server**, **port**, **admindatabase, databaseadminusername, database** and **thingworxusername** can be configured in the script, depending on the requirements. Execute this script with the **--help** option for usage information.

   As an example, to set up the database and a postgres user name, assuming the user created above is **twadmin**, enter:

   $ sudo sh thingworxPostgresDBSetup.sh –h rds-host –d thingworx –b postgres –a pgadmin –u twadmin

   where **twadmin** is the user name and **pgadmin** is the administrator account created during RDS installation

4. Execute the script. Once executed, this creates a new database in the Amazon RDS PostgreSQL instance.

thingworxRdsPostgresDBSetup.sh Script Options

Option	Parameter	Default	Description	Example
-h or -H	Server name or IP	rds-host	Hostname or IP of RDS PostgreSQL instance	-h rds-host
-p or -P	port	5432	Port number of PostgreSQL	-p 5432
-d or -D	database	thingworx	PostgreSQL Database name to create	-d thingworx
-b or -B	admindatabase	postgres	Name of the PostgreSQL system database	-b postgres
-a or -A	adminusername	postgres	Administrator Name	-a pgadmin
-u or -U	thingworxusername	twadmin	User name that has permissions to write to the database.	-u twadmin

Configuring and Executing the Model/Data Provider Schema Script (RDS)

To set up the PostgreSQL model/data provider schema, the **thingworxPostgresSchema.sh** script must be configured and executed. This will set up the public schema under your database on the Amazon RDS PostgreSQL instance.

- Configuring and Executing the Model/Data Provider Schema Script (Windows)
- Configuring and Executing the Model/Data Provider Schema Script (Ubuntu)
- Configuring and Executing the Model/Data Provider Schema Script (RHEL)

Configuring platform-settings.json (RDS)

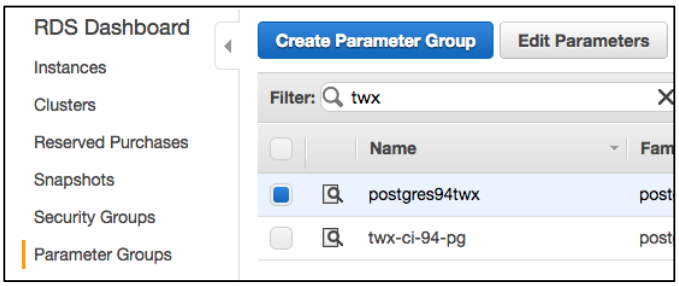
This is identical to the sections above except for the Amazon RDS PostgresSQL host name/ip.

- Configuring platform-settings.json (Windows)
- Configuring platform-settings.json (Ubuntu)
- Configuring platform-settings.json (RHEL)

Installing and Configuring PostgreSQL DB Host Servers (RDS)

The DB host server is the Amazon RDS instance that was created above.

1. Edit the parameter group created earlier.



2. Edit the parameters listed below to suit your environment.

   NOTE: The values listed in the Configuration column reflect the example deployment in the reference architecture, but can be modified for your environment. For many of the settings in the table below, links are provided to help you determine the configuration values for your environment. RDS specific information can be found at - http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.PostgreSQL.CommonDBATasks.html

Setting	Configuration	Description
shared_buffers	1024MB	Optional performance tuning. Sets the amount of memory the database server uses for shared memory buffers. It is recommended to set this at 1/4 of the memory available on the machine. Refer to http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-SHARED-BUFFERS
work_mem	32MB	Optional performance tuning. Specifies the amount of memory to be used by internal sort operations and hash tables before writing to temporary disk files. Refer to http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-WORK-MEM

maintenance_work_mem	512MB	Optional performance tuning. Specifies the maximum amount of memory to be used by maintenance operations. Refer to http://www.postgresql.org/docs/current/static/runtime-config-resource.html#GUC-MAINTENANCE-WORK-MEM
effective_cache_size		Should be set to an estimate of how much memory is available for disk caching by the OS and within the database. It is recommended to set this to half the memory available on the machine.
checkpoint_segments		Depends on the size of the PostgreSQL box. Should set to 32/64/128/256, depending upon machine size.
checkpoint_completion_target		If the **checkpoint_segments** is changed from the default value of 3, change this to 0.9.
ssl_renegotiation_limit		If PostgreSQL is deployed Ubuntu, set this value to 0 (never) or increase the default (512MB) to something larger, e.g. 2GB to avoid ssl renegotiation from happening too often between master and synchronous slave.

Installing ThingWorx (RDS)

Thingworx server installation is identical to the previous section since the only difference is using RDS instead of installing PostgreSQL database on your own. Refer to the relevant OS section to install ThingWorx:

- Windows
- Ubuntu
- RHEL

Appendix A: Tomcat Java Option Settings

Mandatory Settings

Setting	Description
-server	Explicitly tells the JVM to run in server mode. This is true by default when using 64-bit JDK, but it is best practice to declare it.
-d64	Explicitly tells the JVM to run in 64-bit mode. The current JVM automatically detects this, but it is best practice to declare it.
XX:+UseG1GC	Tells the JVM to use the Garbage First Garbage Collector.
-Dfile.encoding=UTF-8	Tells the JVM to use UTF-8 as the default character set so that non-Western alphabets are displayed correctly.
-Xms3072m (for a system with 4GB of memory)	Tells the JVM to allocate a minimum of 3072MB of memory to the Tomcat process. This should be set to 75% of the available system memory. NOTE: The amount of memory needs to be tuned depending on the actual environment.
-Xmx3072m (for a system with 4GB of memory)	Tells the JVM to limit the maximum memory to the Tomcat process. This should be set to 75% of the available system memory. NOTE: The amount of memory needs to be tuned depending on the actual environment. 5GB of memory is a good starting point for 100,000 things. NOTE: The reason that the min and max amounts of memory are made equal is to reduce JVM having to re-evaluate required memory and resizing the allocation at runtime. While this is recommended for hosted and/or public-facing environments, for development and test environments, using **–Xms512m** would suffice. Also, verify that there is enough memory left to allow the operating system to function.

Optional Settings to Enable JMX Monitoring for VisualVM or JConsole

Setting	Description
-Dcom.sun.management.jmxremote	Notifies the JVM that you plan to remote monitor it via JMX
-Dcom.sun.management.jmxremote.port=22222	The port the JVM should open up for monitoring.
-Dcom.sun.management.jmxremote.ssl=false	No SSL usage.
-Dcom.sun.management.jmxremote.authenticate=false	No authentication required.
-Djava.rmi.server.hostname=<host or IP>	The hostname or IP that the underlying RMI client connection will use.

Appendix B: Sample platform-settings.json

The platform-settings.json file is available in the software download.

```
{
      "PlatformSettingsConfig": {
            "BasicSettings": {
                  "BackupStorage": "/ThingworxBackupStorage",
                  "DatabaseLogRetentionPolicy": 7,
                  "EnableBackup": true,
                  "EnableHA": false,
                  "EnableSystemLogging": false,
                  "HTTPRequestHeaderMaxLength": 2000,
                  "HTTPRequestParameterMaxLength": 2000,
                  "Storage": "/ThingworxStorage"
            },

            "HASettings": {
                  "CoordinatorConnectionTimeout": 15000,
                  "CoordinatorHosts": "127.0.0.1:2181",
                  "CoordinatorMaxRetries": 3,
                  "CoordinatorRetryTimeout": 1000,
                  "CoordinatorSessionTimeout": 60000,
                  "LoadBalancerBase64EncodedCredentials":
"QWRtaW5pc3RyYXRvcjphZG1pbg=="
            }
      },

      "PersistenceProviderPackageConfigs": {

            "H2PersistenceProviderPackage": {
                  "ConnectionInformation": {
                        "acquireIncrement": 5,
                        "acquireRetryAttempts": 30,
                        "acquireRetryDelay": 1000,
                        "checkoutTimeout": 2000,
                        "idleConnectionTestPeriod": 6,
                        "initialPoolSize": 10,
                        "maxConnectionAge": 0,
                        "maxIdleTime": 0,
                        "maxIdleTimeExcessConnections": 36000,
                        "maxPoolSize": 100,
                        "maxStatements": 0,
                        "maxStatementsPerConnection": 50,
```

```
                "minPoolSize": 10,
                "numHelperThreads": 6,
                "tableLockTimeout": 10000,
                "testConnectionOnCheckout": false,
                "unreturnedConnectionTimeout": 0
        },

        "StreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 250000,
                "maximumWaitTime": 10000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
        },

        "ValueStreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumWaitTime": 10000,
                "maximumQueueSize": 500000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
        }
},
"NeoPersistenceProviderPackage": {
        "StreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 250000,
                "maximumWaitTime": 10000,
                "scanRate": 5,
                "sizeThreshold": 1000
        },

        "ValueStreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 500000,
                "maximumWaitTime": 10000,
                "scanRate": 5,
                "sizeThreshold": 1000
        }
},
```

```
            "PostgresPersistenceProviderPackage": {
                  "ConnectionInformation": {
                        "acquireIncrement": 5,
                        "acquireRetryAttempts": 3,
                        "acquireRetryDelay": 10000,
                        "checkoutTimeout": 1000000,
                        "driverClass": "org.postgresql.Driver",
                        "fetchSize": 5000,
                        "idleConnectionTestPeriod": 60,
                        "initialPoolSize": 5,
                        "jdbcUrl":
"jdbc:postgresql://localhost:5432/thingworx",
                        "maxConnectionAge": 0,
                        "maxIdleTime": 0,
                        "maxIdleTimeExcessConnections": 300,
                        "maxPoolSize": 100,
                        "maxStatements": 100,
                        "minPoolSize": 5,
                        "numHelperThreads": 8,
                        "password": "password",
                        "testConnectionOnCheckout": false,
                        "unreturnedConnectionTimeout": 0,
                        "username": "twadmin"
                  },

                  "StreamProcessorSettings": {
                        "maximumBlockSize": 2500,
                        "maximumQueueSize": 250000,
                        "maximumWaitTime": 10000,
                        "numberOfProcessingThreads": 5,
                        "scanRate": 5,
                        "sizeThreshold": 1000
                  },

                  "ValueStreamProcessorSettings": {
                        "maximumBlockSize": 2500,
                        "maximumQueueSize": 500000,
                        "maximumWaitTime": 10000,
                        "numberOfProcessingThreads": 5,
                        "scanRate": 5,
                        "sizeThreshold": 1000
                  }
            }
```

```
        }
}
```

Appendix C: platform-settings.json Options

NOTE: The platform-settings.json file is available for administrators to adjust settings for fine-tuning.

platform-settings.json Options		
Setting	Default	Description
Core Platform Settings		
BackupStorage	/ThingworxBackupStorage	The directory name where all backups are written to.
DatabaseLogRetentionPolicy	7	The number of days that database logs are retained.
EnableBackup	true	Determines whether backups are retained.
EnableHA	false	Determines whether ThingWorx can be configured for a highly available landscape.
EnableSystemLogging	false	Determines whether system logging is enabled. NOTE: DO NOT TURN THIS ON UNLESS INSTRUCTED BY THINGWORX SUPPORT.
HTTPRequestHeaderMaxLength	2000	The maximum allowable length for HTTP Request Headers values.
HTTPRequestParameterMaxLength	2000	The maximum allowable length for HTTP Request Parameter values.
Storage	/ThingworxStorage	The directory where all storage directories are created/located (excluding Backup Storage).
HA Settings Settings specific to a PostgreSQL HA landscape configuration. All are optional, and are ignored if the **EnableHA** setting above is set to **false**.		
CoordinatorConnectionTimeout	15000	How long to wait (in milliseconds) for a connection to be established with process/server used to coordinate ThingWorx leadership.
CoordinatorHosts	127.0.0.1:2181	A comma-delimited list of server IP addresses on which the processes used to coordinate ThingWorx leadership exist (e.g. "127.0.0.1:2181, 127.0.0.2:2181").
CoordinatorMaxRetries	3	The maximum allowable number of retries that will be made to establish a connection with the process/server used to coordinate ThingWorx leadership.
CoordinatorRetryTimeout	1000	How long to wait (in milliseconds) for each retry attempt.
CoordinatorSessionTimeout	60000	How long ThingWorx waits (in milliseconds) without receiving a "heartbeat" from the process/server used to coordinate ThingWorx leadership.

LoadBalancerBase64EncodedCredentials	QWRtaW5pc3RyYXRvcjphZG1pbg=="	The Base64-encoded credentials for the HA Load Balancer, in the format of *<user>:<password>*. NOTE: You can use any utility that Base64 encodes the matching *<user>:<password>* string used in your load balancer setup.
PostgresPersistenceProviderPackage PostgreSQL-specific persistence provider settings. If PostgreSQL is not the persistence provider, then this entire section should be ignored.		
acquireIncrement	5	Determines how many connections at a time the platform will try to acquire when the pool is exhausted.
acquireRetryAttempts	3	Defines how many times ThingWorx will try to acquire a new Connection from the database before giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
checkoutTimeout	10000000	The number of milliseconds a client calling getConnection() will wait for a Connection to be checked-in or acquired when the pool is exhausted.
driverClass	org.postgresql.Driver	The fully-qualified class name of the JDBC driverClass that is expected to provide Connections.
fetchSize	5000	The count of rows to be fetched in batches instead of caching all rows on the client side.
idleConnectionTestPeriod	60	If this is a number greater than 0, ThingWorx will test all idle, pooled but unchecked-out connections, every x number of seconds.
initialPoolSize	5	Initial number of database connections created and maintained within a pool upon startup. Should be between minPoolSize and maxPoolSize.

jdbcUrl	jdbc:postgresql://localhost:5432/thingworx”	The jdbc url used to connect to PostgreSQL. NOTE: If the default schema name is changed (from public), you must add <databasename>?currentSchema=<name of schema> For example, if the schema name is mySchema, it would be: **jdbc:postgresql://<*DBServer*>:<*DBPort*>/<*databasename*>?currentSchema=mySchema** NOTE: If you are configuring an HA solution, this should reflect the server IP that the pgPool process is running on. Change the port to the port that pgPool is serving.
maxConnectionAge	0	Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool.
maxIdleTime	0	Seconds a connection can remain pooled but unused before being discarded. Zero means idle connections never expire.
maxIdleTimeExcessConnections	300	The number of seconds that connections in excess of minPoolSize are permitted to remain in idle in the pool before being culled. Intended for applications that wish to aggressively minimize the number of open connections, shrinking the pool back towards minPoolSize if, following a spike, the load level diminishes and Connections acquired are no longer needed. If maxIdleTime is set, maxIdleTimeExcessConnections should be smaller to have any effect. Setting this to zero means no enforcement and excess connections are not idled out.

maxPoolSize	100	Maximum number of Connections a pool will maintain at any given time.
maxStatements	100	The size of ThingWorx's global PreparedStatement cache.
minPoolSize	5	Minimum number of Connections a pool will maintain at any given time.
numHelperThreads	8	The number of helper threads to spawn. Slow JDBC operations are generally performed by helper threads that don't hold contended locks. Spreading these operations over multiple threads can significantly improve performance by allowing multiple operations to be performed simultaneously.
password	password	The password used to log into the database.
testConnectionOnCheckout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.
unreturnedConnectionTimeout	0	The number of seconds to wait for a response from an unresponsive connection before discarding it. If set, if an application checks out but then fails to check-in a connection within the specified period of time, the pool will discard the connection. This permits applications with occasional connection leaks to survive, rather than eventually exhausting the Connection pool. Zero means no timeout, and applications are expected to close their own connections.
username	twadmin	The user that has the privilege to modify tables. This is the user created on the database for the ThingWorx server.
Stream Processor Settings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that)

maximumWaitTime	10000	Number of milliseconds the system waits before flushing the stream buffer.
numberOfProcessingThreads	5	The number of processing threads (cannot change for Neo4j).
scanRate	5	The buffer status is checked at the specified rate value in milliseconds.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the stream buffer.
Value Stream Processor Settings		
maximumBlockSize	2500	Maximum number of value stream writes to process in one block.
maximumQueueSize	500000	Maximum number of value stream entries to queue (will be rejected after that).
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the value stream buffer.
numberofProcessingThreads	5	The number of processing threads (cannot change for Neo4j).
scanRate	5	The rate (in milliseconds) before flushing the stream buffer.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the value stream buffer.
NeoPersistenceProviderPackage Contains Neo4j-specific Persistence Provider settings. If Neo is not the Persistence Provider, then this entire section should be ignored.		
StreamProcessorSettings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing stream buffer.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.
V**alueStreamProcessorSettings**		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.

maximumQueueSize	500000	The maximum number of stream entries to queue (will be rejected after that).
maximumWaitTime	10000	The maximum wait time (in milliseconds) before flushing the stream buffer.
scanRate	5	The rate (in milliseconds) at which to check the buffer status.
sizeThreshold	1000	The maximum number of items to accumulate before flushing stream buffer.

Appendix D: Metrics Reporting

You can opt in to allow your ThingWorx Core metrics data (such as usage, performance, and diagnostics) to be sent to a PTC server. The configuration settings for metrics reporting are included in the Platform Subsystem. For more information, see the Platform Subsystem topic in the Help Center.

Appendix E: Installing PostgreSQL Client Package and PostgreSQL User

In order to issue PostgreSQL commands from the client machine to the PostgreSQL server, do so from a PostgreSQL user. To do so, the postgresql-client-9.4 package can be installed on the client machine, refer to your distributions documentation on how to install it. This package provides some administration tools such as **psql** that are discussed later in the monitoring/administration section

Windows PostgreSQL client comes along with PgAdmin lll. Follow the same instructions for installation.
Ubuntu `sudo apt-get install postgresql-client`
RHEL `sudo yum install postgresql94.x86_64`