Building ThingWorx 8.3 Docker Images

V1.0

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Contents

1

ThingWorx Docker Overview

ThingWorx Docker is a set of Dockerfiles and supporting scripts to enable building Docker Images of the ThingWorx Platform. These images can then be run locally on a developer machine or in a container orchestration platform such as Kubernetes.

The ThingWorx Platform Dockerfiles archive contains a build script (`build.sh`) and a variable file (`build.env`) to help simplify the image building process. These are covered in Setting Up ThingWorx Docker on page 6.

The release also includes the `dockerfiles` folder ,where the actual Dockerfiles and scripts for each ThingWorx Platform content provider are located. PTC provides a set of Docker files and supplemental scripts to build images for the following content providers:

- H2
- PostgreSQL
- Microsoft SQL Server

Where to Get ThingWorx Docker

You can download ThingWorx Platform Dockerfiles from the **ThingWorx Platform** part of the PTC Software Downloads site. The archive is available for ThingWorx versions 8.3 and later, and is listed under **ThingWorx Dockerfiles**.

Customized ThingWorx Docker Images

The default Dockerfiles and scripts are intended to make it easier to deploy and run the ThingWorx Platform. For some use cases, you might need to modify and customize the Dockerfiles and supporting scripts to fit your environment. In these cases, the resulting ThingWorx Images are not supported by PTC.

Prerequisites

This section covers the supported operating systems and required Docker software to run ThingWorx Docker.

Operating System

The following operating systems are supported.

- Microsoft Windows 10
- Red Hat Enterprise Linux (RHEL) 7 Update 1
- Amazon EC2 Linux (64-bit)
- Ubuntu 16.02

Note

You must use the 64-bit versions.

Docker Versions

The following Docker versions are required:

- Docker Community Edition (docker-ce)

  Release 18.05.0-ce is recommended. To install the Docker Community Edition on your system, follow the instructions for your operating system on the Docker web site: https://www.docker.com/community-edition#/download.

- Docker Compose (docker-compose)

  Release 1.17.1 is recommended. To install the Docker Compose on your system, follow the instructions for your operating system on the Docker web site: https://docs.docker.com/compose/install/.

2

Setting Up ThingWorx Docker

This chapter covers how to set up and build ThingWorx Docker images. It also covers configuring ThingWorx Docker.

Setting Up ThingWorx Docker Builds

To build ThingWorx Docker images you perform both of the following actions:

- Make sure the required binaries are staged and available for the build process.
- Modify the `build.env` variable file with appropriate values.

Required Files

The following files are required to build ThingWorx Docker.

- Required files for all platform versions:
  - `template-processor` – A tool provided by PTC to both parse templates inside the Docker container when it is starting to provide variables and format configuration files based on the running environment.
    Example file: `template-processor-12.0.0.9-application.tar.gz`
  - `tomcat` – The Tomcat archive file obtained from Apache to run the ThingWorx Platform.
    Example file: `tomcat-9.0.21.tar.gz`
  - `java` – The Java JDK (version 8) archive file obtained from Oracle.
    Example file: `jdk-8u172-linux-x64.tar.gz`
- ~~Required archive files for the individual platform versions:~~
  - ~~ThingWorx Platform H2~~
    ~~Example file: `Thingworx-Platform-H2-8.3.2-b535.zip`~~
  - ThingWorx Platform PostgreSQL
    Example file: `Thingworx-Platform-Postgres-8.3.2-b535.zip`
  - ~~ThingWorx Platform Microsoft SQL Server~~
    ~~Example file: `Thingworx-Platform-Mssql-8.3.2-b535.zip`~~
    ~~This version also requires the Microsoft JDBC Driver for SQL Server from Microsoft.~~
    ~~Example file: `sqljdbc_6.0.8112.200_enu.tar.gz`~~

`build.env` Variables

The `build.env` file contains a list of variables that you must set. The following variables must be set:

Variable Name	Default	Comment
JAVA_VERSION	8u172	Specifies the version of the Oracle Java JDK.
TOMCAT_VERSION	9.0.21	Specifies the version of Apache Tomcat.
TEMPLATE_PROCESSOR_VERSION	12.0.0.9	Specifies the version of the template-processor archive file in the `staging` folder.
PLATFORM_SETTINGS_FILE	platform-settings.json	Specifies the path to a base ThingWorx settings file (included in the `staging` folder).
BUILD_TEST_DBS	true	Determines whether to build database images for testing, alongside the platform images. Set to `true` to build the database images.
~~PLATFORM_H2_VERSION~~	8.3.2-b535	Specifies the version of the ThingWorx H2 Platform to build. Only required when building H2 containers.
~~PLATFORM_H2_ARCHIVE~~	Thingworx-Platform-H2-8.3.2-b535.zip	Specifies the file name of the ThingWorx H2 `.zip` file in the `staging` folder. Only required when building H2 containers.
PLATFORM_POSTGRES_VERSION	8.3.2-b535	Specifies the version of the ThingWorx PostgreSQL Platform to build. Only required when building PostgreSQL containers.
PLATFORM_POSTGRES_ARCHIVE	Thingworx-Platform-Postgres-8.3.2-b535.zip	Specifies the file name of the ThingWorx PostgreSQL `.zip` file in the `staging` folder. Only required when building PostgreSQL containers.
~~PLATFORM_MSSQL_VERSION~~	8.3.2-b535	Specifies the version of the ThingWorx Microsoft SQL Server Platform to build. Only required when building Microsoft SQL Server containers.
~~PLATFORM_MSSQL_ARCHIVE~~	Thingworx-	Specifies the file name of the ThingWorx Microsoft SQL Server

Variable Name	Default	Comment
	Platform-Mssql-8.3.2-b535.zip	`.zip` file in the `staging` folder. Only required when building Microsoft SQL Server containers.
~~SQLDRIVER_VERSION~~	6.0.8112.-200	Specifies the version to install of the Microsoft JDBC Driver for SQL Server. Only required when building Microsoft SQL Server containers.
~~MSSQL_DB_TWX_DATABASE_PASSWORD~~	No default – must be set manually	Specifies the password for the Microsoft SQL Server database user. If BUILD_TEST_DBS is enabled and Microsoft SQL Server images are being built, this must be set for use by the image build process.
~~MSSQL_DB_TWX_DATABASE_USERNAME~~	No default – must be set manually	Specifies the user name for the Microsoft SQL Server database user. If BUILD_TEST_DBS is enabled and Microsoft SQL Server images are being built, this must be set for use by the image build process.
~~MSSQL_DB_TWX_DATABASE_SCHEMA~~	No default – must be set manually	Specifies this is the database schema for the Microsoft SQL Server database. If BUILD_TEST_DBS is enabled and Microsoft SQL Server images are being built, this must be set for use by the image build process.
~~MSSQL_DB_SA_PASSWORD~~	No default – must be set manually	Specifies the password for the Microsoft SQL Server sa user. If BUILD_TEST_DBS is enabled and Microsoft SQL Server images are being built, this must be set for use by the image build process.

The following variables must be set only if the default values do not match the files in the `staging` folder:

Variable Name	Default	Comment
TOMCAT_ARCHIVE	tomcat-*TOMCAT_VERSION*.tar.gz	Specifies the name of the Tomcat archive file in the `staging` folder.
JAVA_ARCHIVE	jdk-*JAVA_VERSION*-linux-x64.tar.gz	Specifies the name of the Java archive file in the `staging` folder.
~~SQLDRIVER_ARCHIVE~~	sqljdbc_*SQLDRIVER_VERSION*_enu.tar.gz	Specifies the name of the Microsoft JDBC Driver for SQL Server archive file in the `staging` folder. Only required when building Microsoft SQL Server containers.
TEMPLATE_PROCESSOR_ARCHIVE	template-processor-*TEMPLATE_PROCESSOR_VERSION*-application.tar.gz	Specifies the name of the `template-processor` archive file in the `staging` folder.

Staging Files

You must put the required files for building the Docker images in the `staging` folder that is part of this release. The `staging` folder should already contain a base `platform-settings.json` file.

To assist with staging, Apache Tomcat and a specific version of the Microsoft JDBC Driver for SQL Server (the default version) can be downloaded automatically.

To download automatically:

1. Make sure you have set the `build.env` file variables appropriately.R
2. un the command `./build.sh stage`.

   If there are no errors, the files should be in the `staging` folder and they should match your `build.env` settings.

You can get the other required files in the following ways:

- Java

You must download Java manually from Oracle due to the requirement to accept Oracle's licensing terms. It is available on the Java SE Development Kit 8 Downloads page. After accepting the license agreement on the page, download the **Linux x64** `tar.gz` file (for example: **jdk-8u181-linux-x64.tar.gz**).

Save this file in the `staging` folder and make sure the JAVA_VERSION and JAVA_ARCHIVE variables in `build.env` file are correct.

- ThingWorx Platform archive files
  You can download the ThingWorx Platform archive files from same **ThingWorx Platform** part of the PTC Software Downloads site that contains this Dockerfile release. Make sure to use same ThingWorx version as that for the set of Dockerfiles, as there could be differences. Example File: `Thingworx-Platform-H2-8.3.2-b535.zip`
  Save this file in the `staging` folder and make sure the PLATFORM_*_VERSION and PLATFORM_*_ARCHIVE variables in `build.env` file are correct.
- Template Processor archive file
  The `template-processor` tool is included in the `staging` folder by default and should be included in the Docker builds automatically. Make sure the TEMPLATE_PROCESSOR_VERSION and TEMPLATE_PROCESSOR_ARCHIVE `build.env` file settings are correct.
- Tomcat archive file
  If you could not automatically download Tomcat, download it directly from Apache at the Tomcat 9 Software Downloads site. Choose to download the **Core** version and select **tar.gz**. Example File: `apache-tomcat-9.0.21.tar.gz`
  Save this file in the `staging` folder and make sure the TOMCAT_VERSION and TOMCAT_ARCHIVE variables in `build.env` file are correct.
- ~~Microsoft JDBC Driver for SQL Server~~
  ~~If you could not automatically download the Microsoft JDBC Driver, or you want to use an alternate version, download it directly from Microsoft at Microsoft JDBC Driver 6.0 for SQL Server. Select the **English** version (as the file structure differs with alternate languages). On the following page, select **sqljdbc_*version*_enu.tar.gz** and click **Next**.~~
  ~~Save this file in the `staging` folder and make sure the SQLDRIVER_VERSION and SQLDRIVER_ARCHIVE variables in `build.env` file are correct.~~

Building the ThingWorx Docker Images

After completing the setup, you can use the build script to create the ThingWorx Docker images. The included `build.sh` script take the variables set in the `build.env` file and works with the files in the `staging` folder to make sure the Docker build command has the appropriate variables and build context available.

To build the images run the following command:

`./build.sh` *type*

type can be one of the following values:

- `h2`
- `postgres`
- `mssql`
- `all`

> **Note**
>
> Choosing `all` builds all of the content providers, which can take some time to complete.

After the build process completes, the following Docker images are available, depending on the content provider you built:

- ~~H2~~

  ~~Platform Docker image:~~ `thingworx/platform-h2:latest`
- PostgreSQL

  Platform Docker image: `thingworx/platform-postgres:latest`

  PostgreSQL DB Docker image: `thingworx/postgres-db:latest`
- ~~Microsoft SQL Server~~

  ~~Platform Docker image:~~ `thingworx/platform-mssql:latest`

  ~~Microsoft SQL Server DB Docker image:~~ `thingworx/mssql-db:latest`

> **Note**
>
> The DB Docker images for PostgreSQL and Microsoft SQL Server are not intended for production use. They are for testing and development only.

Running the ThingWorx Docker Images

This release contains some basic Docker Compose files intended to help you test the built ThingWorx images. To start the compose environment, enter the following command:

```
docker-compose -f docker-compose-type.yml up -d
```

type can be one of the following values:

- `h2`
- `postgres`
- `mssql`

For example, enter these commands to perform the following operations:

- Start an H2 image:

  ```
  docker-compose -f docker-compose-h2.yml up -d
  ```

- View the logs:

  ```
  docker-compose -f docker-compose-h2.yml logs -f
  ```

  Enter CTRL+C to close the logs.

- Stop the image:

  ```
  docker-compose -f docker-compose-h2.yml down
  ```

Configuring ThingWorx Docker

To get up and running quickly with ThingWorx Docker, use the default settings for your content provider. You can also configure the Docker Compose file to customize the settings for your site. This section describes how to configure the Docker Compose files for each content provider.

Configuring HTTP Secure (HTTPS) and HTTP

By default, HTTP is enabled and HTTPS is disabled in the provided ThingWorx images. This is useful for testing and development, but not for use in production. You can optionally disable HTTP and enable HTTPS on the container.

Note

You must provide a keystore to start the container if HTTPS is enabled. Otherwise, the container exits.

To configure HTTPS for the Docker images:

1. Copy the keystore file containing your HTTPS certificates to the Docker mounted `ThingworxPlatform` folder.

   By default, the keystore must be named `keystore.jks`. You can override this with the SSL_KEYSTORE_FILENAME environment variables described later.

   For the example Docker Compose files, copy the file here:

   `./thingworx-storage/shared/ThingworxPlatform`
2. Make sure the Docker Compose file has an environment variable `SSL_KEYSTORE_PASSWORD` set to the encryption password of your keystore and `HTTPS_ENABLED` is set to `true`.
3. Make sure port `8443:8443` is in the `ports` section of your Docker Compose file.

To enable HTTP:

1. In the Docker Compose file, make sure an environment variable `HTTP_ENABLED` is set to `true` or `false`, depending on your requirements.

   The default is `true`.
2. Make sure port `8080:8080` is in the `ports` section of your Docker Compose file.

Note

Where *PASSWORD* appears in the files, you must replace it with the password for ThingWorx or for the database.

H2

The following example is a Docker Compose file for H2 with some basic settings:

```
version: '2.2'

services:
  platform:
    image: thingworx/platform-h2:latest
    healthcheck:
      test: curl -s -w '%{http_code}' -U 'bad:creds'
            localhost:8080/Thingworx/Subsystems/PlatformSubsystem | grep -w 401
      interval: 15s
    ports:
      - "8080:8080"
      - "8443:8443"
```

```
    environment:
      - "INITIAL_HEAP=2"
      - "MAX_HEAP=4"
      - "ENABLE_HTTP=true"
      - "ENABLE_HTTPS=false"
      # NOTE: If supplying a keystore for SSL, you must set your keystore
password
      #- "SSL_KEYSTORE_PASSWORD="
      # NOTE: The following must be set for the Platform to start. This will be
      # the initial Administrator password.
      - "THINGWORX_INITIAL_ADMIN_PASSWORD="
    volumes:
      - "./thingworx-h2-storage/ThingworxPlatform:/ThingworxPlatform"
      - "./thingworx-h2-storage/ThingworxStorage:/ThingworxStorage"
      - "./thingworx-h2-storage/ThingworxBackupStorage:/ThingworxBackupStorage"
      - "./thingworx-h2-storage/tomcat-logs:/opt/apache-tomcat/logs"
```

Note

The initial memory is set to 2 GB, and the maximum memory is set to 4 GB. The volume mounts are relative to docker-compose files. This is the location where the logs and configuration files are stored to enable persistence in the containers.

You can add the following options to the `environment` section of the file to control the configuration of this instance:

Variable Names	Values	Defaults	Comments
INITIAL_HEAP	Number	2	Sets the initial memory size for the instance in GB.
MAX_HEAP	Number	4	Sets the maximum memory size for the instance in GB
ENABLE_HTTP	true/false	true	Enables the HTTP connector on Tomcat for unsecured traffic to the container.
ENABLE_HTTPS	true/false	false	Enables the HTTPS connector on Tomcat for secured traffic to the container. You must also provide a keystore and SSL_KEYSTORE_PASSWORD must be set.
SSL_KEYSTORE_PASSWORD	String	*PASSWORD*	Sets the password to the keystore used for SSL communication in Tomcat.
SSL_KEYSTORE_	String	/Thing-	Sets the path to the Tomcat SSL

Variable Names	Values	Defaults	Comments
BASE_PATH		worxPlat-form	keystore. If you store the keystore in a location in the container other than the default folder, you must set this variable.
SSL_KEYSTORE_ FILENAME	String	keystore. jks	Sets the file name for the Tomcat SSL keystore. If your keystore has a file name other than the default, you must set this variable.
TOMCAT_SSL_ PROTOCOLS	String	TLSv1.2	Specifies the Tomcat SSL protocol. Set this if you want to override the accepted SSL protocols in Tomcat.
SERVER_HTTP_ PORT	String	8080	Specifies the port that Tomcat monitors for HTTP communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
SERVER_HTTPS_ PORT	String	8443	Specifies the port that Tomcat monitors for HTTPS communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
DOCKER_DEBUG	true/false	false	Toggles the option for recording debugging information when the container starts up. Note that this might contain sensitive information.
LS_USERNAME	String	""	Specifies your PTC login user name to get your ThingWorx License.
LS_PASSWORD	String	""	Specifies your PTC login password to get your ThingWorx License.
ENABLE_BACKUP	true/false	false	Toggles the option for back ups.
ENABLE_LOGGING	true/false	true	Toggles the option for logging.
ENCRYPT_ CREDENTIALS	true/false	false	Toggles the option to encrypt passwords for databases and licensing in the `platform-settings.json` file.

You can also change the volume path to a location specific to your site. The volume path uses the following syntax:

"*./path_to_local_mount:/path_to_container_mount*"

For example:

```
"/opt/ThingworxPlatform:/ThingworxPlatform"
```

Note

When updating the volume path change only the local mount, as the mount points of the internal container never change.

Microsoft SQL Server

The following example is a Docker Compose file for Microsoft SQL Server with some basic settings:

```
version: '2.2'

services:
  mssql:
    image: thingworx/mssql-db:latest
    ports:
      - "1433"
    healthcheck:
      test: /opt/mssql-tools/bin/sqlcmd -U SA -P "$${SA_PASSWORD}" -h -1
            -Q "set nocount on; select serverproperty('ServerName')" | grep -w "$${HOSTNAME}"
      interval: 15s

  platform:
    image: thingworx/platform-mssql:latest
    healthcheck:
      test: curl -s -w '%{http_code}' -U 'bad:creds'
            localhost:8080/Thingworx/Subsystems/PlatformSubsystem | grep -w 401
      interval: 15s
    depends_on:
      mssql:
        condition: service_healthy
    ports:
      - "8080:8080"
      - "8443:8443"
    environment:
      - "INITIAL_HEAP=2"
      - "MAX_HEAP=4"
      - "DATABASE_HOST=mssql"
      - "DATABASE_PORT=1433"
      - "TWX_DATABASE_USERNAME=thingworx"
      - "TWX_DATABASE_SCHEMA=thingworx"
      # NOTE: TWX_DATABASE_PASSWORD for MSSQL platform must be set to match your
```

```
      # environment, or the MSSQL_DB_TWX_DATABASE_PASSWORD that you set for the
      # Test MSSQL DB.
      - "TWX_DATABASE_PASSWORD="
      # NOTE: The following must be set for the Platform to start. This will be
      # the initial Administrator password.
      - "THINGWORX_INITIAL_ADMIN_PASSWORD="
      - "ENABLE_HTTP=true"
      - "ENABLE_HTTPS=false"
      # NOTE: If supplying a keystore for SSL, you must set your keystore password
      #- "SSL_KEYSTORE_PASSWORD="
    volumes:
      - "./thingworx-mssql-storage/ThingworxPlatform:/ThingworxPlatform"
      - "./thingworx-mssql-storage/ThingworxStorage:/ThingworxStorage"
      - "./thingworx-mssql-storage/ThingworxBackupStorage:/ThingworxBackupStorage"
      - "./thingworx-mssql-storage/tomcat-logs:/opt/apache-tomcat/logs"
```

Note

The initial memory is set to 2 GB and the maximum memory is set to 4 GB. The volume mounts are relative to docker-compose files. This is the location where the logs and configuration files are stored to enable persistence in the containers.

A build folder, located in the same folder as the `docker-compose.yml` file, is used to build the default database.

You can add the following options to the `environment` section of the file to control the configuration of this instance:

Variable Names	Values	Defaults	Comments
INITIAL_HEAP	Number	2	Sets the initial memory size for the instance in GB.
MAX_HEAP	Number	4	Sets the maximum memory size for the instance in GB.
DATABASE_HOST	String	mssql	Specifies the hostname, service name, or IP address of the SQL Server database host.
DATABASE_PORT	Number	1433	Specifies the port number for the SQL Server database.
DATABASE_ADMIN_USERNAME	String	SA	Specifies the administrator user name for the SQL Server database.
DATABASE_ADMIN_PASSWORD	String	*PASSWORD*	Specifies the administrator password for the SQL Server database.
ENABLE_HTTP	true/false	true	Enables the HTTP connector on

Variable Names	Values	Defaults	Comments
			Tomcat for unsecured traffic to the container
ENABLE_HTTPS	true/false	false	Enables the HTTPS connector on Tomcat for secured traffic to the container. You must also provide a keystore and SSL_KEYSTORE_PASSWORD must be set.
SSL_KEYSTORE_PASSWORD	String	*PASSWORD*	Specifies the password to keystore used for SSL communication in Tomcat.
SSL_KEYSTORE_BASE_PATH	String	/ThingworxPlatform	Specifies the path to the Tomcat SSL keystore. If you store the keystore in a location in the container other than the default folder, you must set this variable.
SSL_KEYSTORE_FILENAME	String	keystore.jks	Specifies the file name for the Tomcat SSL keystore. If your keystore has a file name other than the default, you must set this variable.
TOMCAT_SSL_PROTOCOLS	String	TLSv1.2	Specifies the Tomcat SSL protocol. Set this if you want to override the accepted SSL protocols in Tomcat.
SERVER_HTTP_PORT	String	8080	Specifies the port that Tomcat monitors for HTTP communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
SERVER_HTTPS_PORT	String	8443	Specifies the port that Tomcat monitors for HTTPS communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
DOCKER_DEBUG	true/false	false	Toggles the option for recording debugging information when the container starts up. Note that this might contain sensitive information.
TWX_DATABASE_USERNAME	String	thingworx	Specifies the ThingWorx user that will be created for the database.
TWX_DATABASE_PASSWORD	String	*PASSWORD*	Specifies the password for the ThingWorx user for the database.

Variable Names	Values	Defaults	Comments
TWX_DATABASE_SCHEMA	String	thingworx	Specifies the schema name for the ThingWorx instance.
LS_USERNAME	String	""	Specifies the PTC login user name to get your ThingWorx License.
LS_PASSWORD	String	""	Specifies your PTC login password to get your ThingWorx License.
ENABLE_BACKUP	true/false	false	Toggles the option for back ups.
ENABLE_LOGGING	true/false	true	Toggles the option for logging.
ENCRYPT_CREDENTIALS	true/false	false	Toggles the option to encrypt passwords for databases and licensing in the `platform-settings.json` file.

You can also change the volume path to a location specific to your site. The volume path uses the following syntax:

"*./path_to_local_mount:/path_to_container_mount*"

For example:

"/opt/ThingworxPlatform:/ThingworxPlatform"

Note

When updating the volume path change only the local mount, as the mount points of the internal container never change.

This compose example uses a Docker image with a pre-loaded ThingWorx database schema. This is useful for testing and development, but should not be used for production.

To connect to a non Docker database, remove the `mssql` service from the Compose file and add the following variables to the `platform` environment variables:

- `DATABASE_ADMIN_USERNAME`
- `DATABASE_ADMIN_PASSWORD`

These are the administrator accounts on your Microsoft SQL Server database authorized to create a user, database, or schema that the ThingWorx container loads on the first startup. Note that this is not needed if the `TWX_DATABASE_USERNAME` and `TWX_DATABASE_SCHEMA` variables are set. See Using an External Microsoft SQL Server Database on page 48 for an example Compose file with the platform setup to use an external database.

If you are using an external database, you can manually install the ThingWorx schema and not provide administrator credentials for the ThingWorx Docker container. To do this, set the DATABASE_HOST, DATABASE_PORT, TWX_DATABASE_USERNAME, TWX_DATABASE_PASSWORD, and TWX_DATABASE_SCHEMA variables appropriately.

PostgreSQL

The following example is a Docker Compose file for PostgreSQL with some basic settings:

```
version: '2.2'

services:
  postgresql:
    image: thingworx/postgres-db:latest
    ports:
      - "5432"
    healthcheck:
      test: pg_isready -U postgres
      interval: 15s
    environment:
      - "TWX_DATABASE_USERNAME=thingworx"
      - "TWX_DATABASE_SCHEMA=thingworx"
      # NOTE: You must set the password for the pre-created ThingWorx Schema in
      # this test database manually.
      - "TWX_DATABASE_PASSWORD="
    volumes:
      - "./thingworx-postgres-storage/ThingworxPostgresqlStorage:/ThingworxPostgresqlStorage"
      - "./thingworx-postgres-storage/postgres-data:/var/lib/postgresql/data"

  platform:
    image: thingworx/platform-postgres:latest
    healthcheck:
      test: curl -s -w '%{http_code}' -U 'bad:creds'
            localhost:8080/Thingworx/Subsystems/PlatformSubsystem | grep -w 401
      interval: 15s
    depends_on:
      postgresql:
        condition: service_healthy
    ports:
      - "8080:8080"
      - "8443:8443"
    environment:
      - "ENABLE_CLUSTERED_MODE=false"
      - "INITIAL_HEAP=2"
      - "MAX_HEAP=4"
```

```
      - "DATABASE_HOST=postgresql"
      - "DATABASE_PORT=5432"
      - "TWX_DATABASE_USERNAME=thingworx"
      - "TWX_DATABASE_SCHEMA=thingworx"
      # NOTE: TWX_DATABASE_PASSWORD must be set for your environment, or to match
      # TWX_DATABASE_PASSWORD in the postgresql service above.
      - "TWX_DATABASE_PASSWORD="
      # NOTE: The following must be set for the Platform to start. This will be
      # the initial Administrator password.
      - "THINGWORX_INITIAL_ADMIN_PASSWORD="
      - "ENABLE_HTTP=true"
      - "ENABLE_HTTPS=false"
      # NOTE: If supplying a keystore for SSL, you must set your keystore password
      #- "SSL_KEYSTORE_PASSWORD="
    volumes:
      - "./thingworx-postgres-storage/ThingworxPlatform:/ThingworxPlatform"
      - "./thingworx-postgres-storage/ThingworxStorage:/ThingworxStorage"
      - "./thingworx-postgres-storage/ThingworxBackupStorage:/ThingworxBackupStorage"
      - "./thingworx-postgres-storage/tomcat-logs:/opt/apache-tomcat/logs"
```

Note

The initial memory is set to 2 GB and the maximum memory is set to 4 GB. The volume mounts are relative to docker-compose files. This is the location where the logs and configuration files are stored to enable persistence in the containers.

A build folder, located in the same folder as the `docker-compose.yml` file, is used to build the default database.

You can add the following options to the `environment` section of the file to control the configuration of this instance:

Variable Names	Values	Defaults	Comments
INITIAL_HEAP	Number	2	Sets the initial memory size for the instance in GB.
MAX_HEAP	Number	4	Sets the maximum memory size for the instance in GB.
DATABASE_HOST	String	post-gressql	Specifies the hostname, service name, or IP address of the PostgreSQL database host.
DATABASE_PORT	Number	5432	Specifies the port number for the PostgreSQL database.
DATABASE_ADMIN_	String	postgres	Specifies the administrator user name

Variable Names	Values	Defaults	Comments
USERNAME			for the PostgreSQL database.
DATABASE_ADMIN_PASSWORD	String	*PASSWORD*	Specifies the administrator password for the PostgreSQL database.
ENABLE_HTTP	true/false	true	Enables the HTTP connector on Tomcat for unsecured traffic to the container
ENABLE_HTTPS	true/false	false	Enables the HTTPS connector on Tomcat for secured traffic to the container. You must also provide a keystore and SSL_KEYSTORE_PASSWORD must be set.
SSL_KEYSTORE_PASSWORD	String	*PASSWORD*	Specifies the password to keystore used for SSL communication in Tomcat.
SSL_KEYSTORE_BASE_PATH	String	/ThingworxPlatform	Specifies the path to the Tomcat SSL keystore. If you store the keystore in a location in the container other than the default folder, you must set this variable.
SSL_KEYSTORE_FILENAME	String	keystore.jks	Specifies the file name for the Tomcat SSL keystore. If your keystore has a file name other than the default, you must set this variable.
TOMCAT_SSL_PROTOCOLS	String	TLSv1.2	Specifies the Tomcat SSL protocol. Set this if you want to override the accepted SSL protocols in Tomcat.
SERVER_HTTP_PORT	String	8080	Specifies the port that Tomcat monitors for HTTP communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
SERVER_HTTPS_PORT	String	8443	Specifies the port that Tomcat monitors for HTTPS communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
DOCKER_DEBUG	true/false	false	Toggles the option for recording debugging information when the container starts up. Note that this might contain sensitive information.

Variable Names	Values	Defaults	Comments
TWX_DATABASE_USERNAME	String	thingworx	Specifies the ThingWorx user that will be created for the database.
TWX_DATABASE_PASSWORD	String	*PASSWORD*	Specifies the password for the ThingWorx user for the database.
TWX_DATABASE_SCHEMA	String	thingworx	Specifies the schema name for the ThingWorx instance.
TABLESPACE_LOCATION	String	/ThingworxPostgresqlStorage	Specifies the location of the tablespace on the database server.
IS_RDS	yes/no	no	Toggles the option for connecting to an RDS PostgreSQL database.
LS_USERNAME	String	""	Specifies the PTC login user name to get your ThingWorx License.
LS_PASSWORD	String	""	Specifies the PTC login password to get your ThingWorx License.
ENABLE_BACKUP	true/false	false	Toggles the option for back ups.
ENABLE_LOGGING	true/false	true	Toggles the option for logging.
ENCRYPT_CREDENTIALS	true/false	false	Toggles the option to encrypt passwords for databases and licensing in the `platform-settings.json` file.

You can also change the volume path to a location specific to your site. The volume path uses the following syntax:

"*./path_to_local_mount:/path_to_container_mount*"

For example:

```
"/opt/ThingworxPlatform:/ThingworxPlatform"
```

Note

When updating the volume path change only the local mount, as the mount points of the internal container never change.

This compose example uses a Docker image with a pre-loaded ThingWorx database schema. This is useful for testing and development, but should not be used for production.

To connect to a non Docker database you can remove the `postgresql` service from the Compose file and add the following variables to the `platform` environment variables:

- `DATABASE_ADMIN_USERNAME`
- `DATABASE_ADMIN_PASSWORD`.

These are administrator accounts on your PostgreSQL database authorized to create a user, database, or schema that the ThingWorx container loads on the first startup. This is not needed if the `TWX_DATABASE_USERNAME` and `TWX_DATABASE_SCHEMA` variables are set. See Using an External PostgreSQL Database on page 50 for an example Compose file with the platform setup to use an external database.

If you are using an external database, you can manually install the ThingWorx schema as usual and not provide administrator credentials for the ThingWorx Docker container. To do this, the `DATABASE_HOST`, `DATABASE_PORT`, `TWX_DATABASE_USERNAME`, `TWX_DATABASE_PASSWORD`, and `TWX_DATABASE_SCHEMA` variables must be set appropriately.

~~Configuring ThingWorx High Availability Docker~~

~~The provided Dockerfiles for the PostgreSQL content provider support running ThingWorx in High Availability (HA) mode. However, running in HA mode requires the following additional external services:~~

- ~~Zookeeper~~
- ~~HAProxy~~
- ~~Pgpool (optional)~~

  ~~Pgpool is required if you are also going to configure PostgreSQL in HA mode.~~

~~More information on how to configure ThingWorx HA is available in the ThingWorx 8 High Availability Administrator's Guide on the PTC **Reference Documents** site.~~

~~In the **ThingWorx Dockerfiles** archive beside the build scripts, there is an example implementation of PostgreSQL HA and its required components. This is intended as an example only and is useful for testing or as a starting point. It is not intended for production use. The example compose file is in the folder `examples/compose/postgres-ha`.~~

Note

The initial memory is set to 2 GB, and the maximum memory is set to 4 GB. The volume mounts are relative to docker-compose files. This is the location where the logs and configuration files are stored to enable persistence in the containers.

There is a build folder in the same folder as the `docker-compose.yml` file. This is required to build all of the HA components.

You can add the following options to the `environment:` section of the file to control the configuration of this instance:

Variable Names	Values	Defaults	Comments
INITIAL_HEAP	Number	2	Sets the initial memory size for the instance in GB.
MAX_HEAP	Number	4	Sets the maximum memory size for the instance in GB.
DATABASE_HOST	String	postgresql	Specifies the hostname, service name, or IP address of the PostgreSQL database host.
DATABASE_PORT	Number	5432	Specifies the port number for the PostgreSQL database.
DATABASE_ADMIN_USERNAME	String	postgres	Specifies the administrator user name for the PostgreSQL database.
DATABASE_ADMIN_PASSWORD	String	*PASSWORD*	Specifies the administrator password for the PostgreSQL database.
TWX_DATABASE_USERNAME	String	thingworx	Specifies the ThingWorx user that will be created for the database.
TWX_DATABASE_PASSWORD	String	*PASSWORD*	Specifies the password for the ThingWorx user for the database.
TWX_DATABASE_SCHEMA	String	thingworx	Specifies the schema name for the ThingWorx instance.
TABLESPACE_LOCATION	String	/ThingworxPostgresqlStorage	Specifies the location of the tablespace on the database server.
IS_RDS	yes/no	no	Toggles the option for connecting to an RDS PostgreSQL database.
LS_USERNAME	String	""	Specifies the PTC login user name to get your ThingWorx License.

Variable Names	Values	Defaults	Comments
LS_PASSWORD	String	""	This is your PTC login password to get your ThingWorx License.
COORDINATOR_HOSTS	String	""	Specifies the ZooKeeper configuration for an HA. It contains a comma separated list of *host:port* entries, such as `zoo1:2181,zoo2:2181,zoo3:2181`
ENABLE_BACKUP	true/false	false	Toggles the option for back ups.
ENABLE_HA	true/false	false	Toggles the option for HA.
ENABLE_HTTP	true/false	false	Enables the HTTP connector on Tomcat for unsecured traffic to the container
SSL_KEYSTORE_PASSWORD	String	*PASSWORD*	Specifies the password to keystore used for SSL communication in Tomcat.
SSL_KEYSTORE_BASE_PATH	String	/ThingworxPlatform	Specifies the path to the Tomcat SSL keystore. If you store the keystore in a location in the container other than the default folder, you must set this variable.
SSL_KEYSTORE_FILENAME	String	keystore.jks	Specifies the file name for the Tomcat SSL keystore. If your keystore has a file name other than the default, you must set this variable.
TOMCAT_SSL_PROTOCOLS	String	TLSv1.2	Specifies the Tomcat SSL protocol. Set this if you want to override the accepted SSL protocols in Tomcat.
SERVER_HTTP_PORT	String	8080	Specifies the port that Tomcat monitors for HTTP communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
SERVER_HTTPS_PORT	String	8443	Specifies the port that Tomcat monitors for HTTPS communication. Note that if this port is changed, you must change the exposed ports in the Compose file.
DOCKER_DEBUG	true/false	false	Toggles the option for recording debugging information when the

Variable Names	Values	Defaults	Comments
			container starts up. Note that this might contain sensitive information.
ENABLE_LOGGING	true/false	true	Toggles the option for logging.
ENCRYPT_CREDENTIALS	true/false	false	Toggles the option to encrypt passwords for databases and licensing in the `platform-settings.json` file.

You can also change the volume path to a location specific to your site. The volume path uses the following syntax:

"*./path_to_local_mount:/path_to_container_mount*"

For example:

"/opt/ThingworxPlatform:/ThingworxPlatform"

Note

When updating the volume path change only the local mount, as the mount points of the internal container never change.

3

Using the ThingWorx Docker Compose Examples

This chapter describes how to start, stop, and troubleshoot the ThingWorx Docker Compose examples.

Starting and Stopping ThingWorx Docker

Note

If your Compose files are not named `docker-compose.yml`, you must pass additional arguments to the Docker compose command such as in the following examples:

- `docker-compose -f docker-compose-h2.yml up -d`
- `docker-compose -f docker-compose-h2.yml down -v`

Starting ThingWorx Docker

To start ThingWorx Docker:

1. Change directories to the location of your `docker-compose.yml file`.
2. Open a command prompt and run the `docker-compose up -d` command to build and start your containers.

   It is suggested that you use `-d` in the command to start it as a daemon.

Stopping ThingWorx Docker

To stop ThingWorx Docker:

1. Change directories to the location of your `docker-compose.yml file`.
2. Open a command prompt and run the `docker-compose down` command to build and start your containers:

   `docker-compose down -v`

Troubleshooting ThingWorx Docker

ThingWorx Docker provides logs to help you troubleshoot an issue with your Docker instance. The logs are written to your local drive using the volume mount in the Docker Compose file. To access the logs, go to the mount points on your local drive.

For example, if your `docker-compose.yml` looks like this:

```
volumes:
  - "./thingworx-storage/shared/ThingworxPlatform:/ThingworxPlatform"
  - "./thingworx-storage/platform1/ThingworxStorage:/
ThingworxStorage"
  - "./thingworx-storage/platform1/ThingworxBackupStorage:/
ThingworxBackupStorage"
```

```
      - "./thingworx-storage/platform1/tomcat-logs:/opt/apache-tomcat/
logs"
```

The logs are written to your local system at the follow locations:

```
./thingworx-storage/platform1/ThingworxStorage/logs
./thingworx-storage/platform1/tomcat-logs
```

4

Upgrading to ThingWorx Docker

When upgrading to a newer version of ThingWorx, you have two options for getting new features and enhancements into existing landscapes: in-place upgrades and migrations. For in-place upgrades, you usually do not need to delete the `ThingworxStorage` and `ThingworxBackupStorage` folders or import data and entities after installing ThingWorx. Migration requires more steps, because you must export data and entities to the `ThingworxStorage` folder and then import those entities and data into the new version of ThingWorx.

Note

See the *Upgrading to ThingWorx* guide for a full compatibility list and more detailed instructions.

Migrating from a Physical Database to a Docker Database

If you choose to migrate from a physical database to a Docker database, refer to the Microsoft SQL Server or PostgreSQL documentation for more information on how to migrate your data. It is also recommended that you follow their best practices guide to get the best performance.

Upgrading Docker

If you are currently using a Docker version and need to upgrade to support a ThingWorx configuration, visit www.docker.com for more information on versions, release notes, and guides for the specific version of Docker.

Doing an In-Place Upgrade

To perform an in-place upgrade to ThingWorx Docker:

1. Follow the steps in Setting Up ThingWorx Docker Builds on page 7 to prepare to build the Docker images.
2. Follow the steps in Building the ThingWorx Docker Images on page 12 to build the Docker images.
3. Stop your current ThingWorx instance.
4. Run the database migration scripts as outlined in the *Upgrading to ThingWorx* guide.
5. Configure your Docker Compose file to point to your database as outlined in Configuring ThingWorx Docker on page 13.
6. Start ThingWorx Docker as outlined in Starting and Stopping ThingWorx Docker on page 30.

Doing a Migration Upgrade

To perform a migration upgrade to ThingWorx Docker:

1. Follow the steps in Setting Up ThingWorx Docker Builds on page 7 to prepare to build the Docker images.
2. Follow the steps in Building the ThingWorx Docker Images on page 12 to build the Docker images.
3. Export your data and entities:
   a. In your current ThingWorx Instance, select **Import/Export ▸ To ThingworxStorage** in ThingWorx Composer.
   b. If necessary, click **Include Data**.
   c. Click **Export**.

   Your data and entities are exported to `ThingworxStorage/exports`.
4. Copy the data and entity export files and move them to a safe location.
5. Note any extensions that are in use.

   These are located in `ThingworxStorage/extensions`.
6. Rename the existing license file located in the `ThingworxPlatform` folder.

   This file can be either `license.bin`, `license_capability_response.bin`, or `successful_capability_response.bin` depending on your current version. Note that you can delete the file, but if a login is unsuccessful, the file will need to be recovered.

7. Verify that your PTC Support site username, password, and timeout (optional) are added to the `platform-settings.json` file in the `PlatformSettingsConfig` section:

```
"LicensingConnectionSettings":{
     "username":"PTC Support site user name",
     "password":"PTC Support site password",
     “timeout”:”60”
 }
```

   If these settings are incorrect, or if the server cannot connect, a license request text file named `licenseRequestFile.txt` is created in the `ThingworxPlatform` folder. In this case, a license must be created manually. If it is not created, ThingWorx starts in limited mode. In limited mode you cannot persist licensed entities to the database. Licensed entities are things, mashups, masters, gadgets, users, and persistence providers.

   Refer to the *Installing ThingWorx* guide for more information on getting a disconnected site license through the PTC Support site.

> **Tip**
>
> If you have any questions or need assistance with generating a license when setting up the manual disconnected mode of licensing, open a case with PTC Technical Support.

8. Stop your current ThingWorx instance.
9. If you are using Microsoft SQL Server or PostgreSQL, run the database migration scripts as outlined in the *Upgrading to ThingWorx* guide.
10. Configure your Docker Compose file to point to your database. as outlined in Configuring ThingWorx Docker on page 13.
11. Start ThingWorx Docker as outlined in Starting and Stopping ThingWorx Docker on page 30.
12. Move the exports files back to the `ThingworxStorage/exports` folder.
13. Login to the platform.
14. Import extensions by choosing **Import/Export** ▸ **Import** in ThingWorx Composer.

    Note:

    - Be sure to import the latest versions of the extensions. If you are upgrading to a major version (for example, from 7.x to 8.0) you must import the 8.x

versions of the extensions. Extensions are available in the PTC Marketplace.

- If you are importing **From ThingworxStorage**, select the **Overwrite Collection Permissions and Organizations** option to overwrite the settings on the server with the collection permissions and organizations in the import. If this option is not selected, by default the collection permissions and organizations from the import are merged with those that are already defined on the server.

15. Import your data and entities by choosing **Import/Export ▸ From ThingworxStorage** in ThingWorx Composer.

A

ThingWorx Docker Licensing

You must have a license file for ThingWorx 8.0 and later. There are two ways to get licensing for ThingWorx Docker. You can either authenticate to the PTC Licensing server and automatically download a license, or start the ThingWorx instance in limited mode and obtain a license from PTC Support.

Authentication Method

To use the authentication method:

1. When starting your instance, add the following options to the `environment` section in your `docker-compose.yml` file:

   ```
   environment:
     - LS_USERNAME=${PTCUSERNAME}
     - LS_PASSWORD=${PTCUSERNAME}
   ```

2. Replace `${PTCUSERNAME}` and `${PTCUSERNAME}` with your username and password for the PTC support site.

   This downloads the license file to your `/ThingworxPlatform` folder.

Limited Mode Method

If you do not have a login, you can start the instance in limited mode by not adding any credentials. This generates a `licenseRequestFile.txt` in the `/ThingworxPlatform` folder. In this case, you need to contact PTC Technical Support for your license file. Once you have it, put the license file in the `//ThingworxPlatform` folder and restart the instance with the following command:

`docker restart ${INSTANCE_NAME}`

Replace `${INSTANCE_NAME}` with the name of your ThingWorx Docker instance.

License Troubleshooting

Some possible issues that can require troubleshooting are described in the following table:

Issue	Possible Resolution
Problem deploying `thingworx.war`.	Verify that the `ThingworxStorage/extensions/web-inf` folder contains the licensing libraries (DLL files).
The following error message appears when you deploy ThingWorx: `org.apache.catalina.core.ApplicationContext.log HTMLManager: FAIL - Deploy Upload Failed, Exception: org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException: the request was rejected because its size (90883556) exceeds the configured maximum (52437800) java.lang.IllegalStateException: org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException: the request was rejected because its size (90883556) exceeds the configured maximum (52437800) at org.apache.catalina.connector.Request.parseParts(Request.java:2871`	The maximum file size in the Tomcat web.xml file must be increased (default is 50MB). This file is located at: `<path to Tomcat>\Apache Software Foundation\Tomcat 8.5\webapps\manager\WEB-INF` 1. Open the web.xml. 2. Change the max-file-size and max-request-size to 104857600. 3. Save and close the file. 4. Restart Tomcat.
The following error message appears when you import a PTC licensed extension: `is licensed but cannot find feature in license.bin file`	Go to the Manage Licenses section on the PTC Support site to confirm the correct license file that matches your entitlement. If you need further assistance with your licenses, contact the License Management team.

Issue	Possible Resolution
The following error message appears when you try to undeploy ThingWorx: `FAIL - Unable to delete [<path to Tomcat>\webapps\Thingworx]. The continued presence of this file may cause problems. Due to FlxCore64.dll (<path to Tomcat>\webapps\Thingworx\WEB-INF\extensions\FlxCore64.dll)`	Remove `-Djava.library.path` from Tomcat's Java configuration before undeploying ThingWorx.

An error message, similar to the following one, appears in the `ConfigurationLog.log`:
`2017-03-10 05:56:07.097-0500 [L: ERROR] [O: ] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] **********LICENSING ERROR ANALYSIS`
`2017-03-10 05:56:07.097-0500 [L: ERROR] [O: ] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] /Library/flexs is listed as a java.library.path but it does not exist.`
`/Library/blah is listed as a java.library.path but it does not exist.`
`/Library/zzz is listed as a java.library.path but it does not exist.  No flx dll files found.`
`Is the java.library.path set?`
`2017-03-10 05:56:07.097-0500 [L: ERROR] [O: ] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] **********END LICENSING ERROR ANALYIS`
	The log message verifies if there is an issue with the license file.

An error message, similar to the following one, appears while the platform is starting:
`2017-06-12 11:33:59.204+0530 [L: ERROR] [O: c.t.s.s.l.LicensingSubsystem] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] [message: The size of provided data is incorrect.]`
`2017-06-12 11:33:59.205+0530 [L: ERROR] [O: c.t.s.s.l.LicensingSubsystem] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] ==========`
`2017-06-12 11:33:59.205+0530 [L: ERROR] [O: c.t.s.s.l.LicensingSubsystem] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] Invalid License file: /ThingworxPlatform\license.bin`
`2017-06-12 11:33:59.205+0530 [L: ERROR] [O: c.t.s.s.l.LicensingSubsystem] [I: ] [U: SuperUser] [S: ] [T: localhost-startStop-1] ==========`
`2017-06-12 11:33:59.205+0530 [L: WARN]`
	The license file may have been opened, edited, or saved in a browser. Download the license file again, rename it to `license_capability_response.bin`, and place in `ThingworxPlatform` folder without editing or saving it.

Issue	Possible Resolution
[O: c.t.s.ThingWorxServer] [I:] [U: SuperUser] [S:] [T: localhost-startStop-1] Shutting down the Platform.	

B

Using Security-Enhanced Linux

Security-Enhanced Linux (SELinux) is a set of kernel modifications and user-space tools that are added to various Linux distributions. SELinux architecture enables the separation of the enforcement of security decisions from the security policy, and streamlines the amount of software involved with security policy enforcement. The key concepts underlying SELinux can be traced to several earlier projects by the United States National Security Agency (NSA).

SELinux users and roles do not have to be related to the actual system users and roles. For every current user or process, SELinux assigns a three string context that includes a username, role, and domain (or type). This system is more flexible than normally required: as a rule, most of the real users share the same SELinux username, and all access control is managed through the third tag, the domain. You must configure the circumstances for allowing a process into a certain domain in the policies. Although you can use the `runcon` command to start a process in an explicitly specified context (user, role and domain), SELinux can deny the transition if it is not approved by the policy.

Files, network ports, and other hardware also have a SELinux context that includes a name, role (seldom used), and type. In the case of file systems, mapping between files and the security contexts is referred to as labeling. The labeling is defined in policy files, but you can also adjust it manually without changing the policies. Hardware types are also detailed, for instance, `bin_t` (all files in the folder `/bin`) or `postgresql_port_t` (PostgreSQL port, 5432). You can specify the SELinux context for a remote file system explicitly at mount time.

SELinux adds the `-Z` switch to the shell commands `ls`, `ps`, and some others, allowing the security context of the files or process to be seen.

Typically, policy rules consist of explicit permissions, such as which domains the user must possess to be able to perform certain actions with the given target. Examples of these actions are read, execute, and in the case of network port, bind or connect. More complex mappings that involve roles and security levels are also possible.

A typical policy consists of a mapping (labeling) file, a rule file, and an interface file that define the domain transition. These three files must be compiled together with the SELinux tools to produce a single policy file. The resulting policy file can be loaded into the kernel, making it active. Loading and unloading policies does not require a reboot. The policy files are either developed manually or can be generated from the more user friendly SELinux management tool. These file are typically tested in permissive mode first, where violations are logged but are allowed. You can use the `audit2allow` tool later to produce additional rules that extend the policy to allow all legitimate activities of the application being confined.

Common SELinux Issues

The most common issue with SELinux is that permission is denied, indicating that SELinux is not working. Permission denied error messages appear when are you are trying to access a system object that is not associated with the program or the user credentials you use for accessing that object. These errors are challenging to resolve, however there are tools that can help with troubleshooting.

Installing Setools and Setroubleshoot

To install these tools on your system:

1. Log into your server or desktop using an account granted administrative rights.
2. Open a command shell.
3. Install `setroubleshoot` packages using Yum:

   ```
   yum install setroubleshoot setools
   ```

SELinux Alerts

Use the `sealert` tool to analyze the audit log used by SELinux. This tool scans the log file and the report, and then generates a report with all discovered SELinux issues.

To run `sealert` from the command-line, point it to the SELinux audit log. See the following example:

```
sealert -a /var/log/audit/audit.log
```

A report describes each issue and explains how to resolve it. Following is a truncated example of the output generated by `sealert`:

```
100% done found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------
```

```
SELinux is preventing /usr/sbin/httpd from getattr access on the file
/myapps/app1/html/index.html.

*****  Plugin catchall_labels (83.8 confidence) suggests  ********************

If you want to allow httpd to have getattr access on the index.html file
Then you need to change the label on /myapps/app1/html/index.html
Do
# semanage fcontext -a -t FILE_TYPE '/myapps/app1/html/index.html'
where FILE_TYPE is one of the following: sssd_var_lib_t, public_content_t, anon_
inodefs_t,

[..truncated..]

httpd_sys_ra_content_t, httpd_sys_rw_content_t, httpd_sys_rw_content_t,
httpd_w3c_validator_content_t.
Then execute:
restorecon -v '/myapps/app1/html/index.html'

*****  Plugin catchall (17.1 confidence) suggests  ***************************

If you believe that httpd should be allowed getattr access on the index.html file
by
default. Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep httpd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp
```

The most important part of the report that explains how to resolve the issue appears at the end of each alert. For example, the report above shows the following solution:

```
If you believe that httpd should be allowed getattr access on the
index.html file by
default. Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep httpd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp
```

The proposed solution creates an SELinux policy to be applied to the problematic file. In this example, an HTML file was assigned the wrong SELinux file context.

Following are more examples for potential issues with the ThingWorx Docker containers.

This example illustrates an issue with PostgreSQL. In this case, `sealert` produces the following alert:

```
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------
-------

SELinux is preventing /bin/chown from setattr access on the directory
data.

*****  Plugin catchall_labels (83.8 confidence) suggests
*******************

If you want to allow chown to have setattr access on the data directory
Then you need to change the label on data
Do
# semanage fcontext -a -t FILE_TYPE 'data'
where FILE_TYPE is one of the following: cgroup_t, nfs_t, svirt_home_t,
svirt_sandbox_file_t.
Then execute:
restorecon -v 'data'


*****  Plugin catchall (17.1 confidence) suggests
**************************

If you believe that chown should be allowed setattr access on the data
directory by
default. Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'chown' --raw | audit2allow -M my-chown
# semodule -i my-chown.pp


Additional Information:
Source Context                system_u:system_r:svirt_lxc_net_
t:s0:c459,c1008
Target Context                unconfined_u:object_r:usr_t:s0
Target Objects                data [ dir ]
Source                        chown
Source Path                   /bin/chown
Port                          <Unknown>
```

```
Host                          <Unknown>
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.13.1-166.el7_4.9.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     ip-10-99-0-109.ec2.internal
Platform                      Linux ip-10-99-0-109.ec2.internal
                              3.10.0-693.5.2.el7.x86_64 #1 SMP Fri Oct 20
                              20:32:50 UTC 2017 x86_64 x86_64
Alert Count                   9
First Seen                    2018-04-02 17:08:08 UTC
Last Seen                     2018-04-02 17:08:27 UTC
Local ID                      65bb52e5-bde9-4ec5-ba44-f0752ffef319

Raw Audit Messages
type=AVC msg=audit(1522688907.927:619): avc:  denied  { setattr } for
pid=12142
comm="initdb" name="data" dev="nvme0n1p1" ino=826278009 scontext=system_
u:system_r:
svirt_lxc_net_t:s0:c459,c1008 tcontext=unconfined_u:object_r:usr_t:s0
tclass=dir


type=SYSCALL msg=audit(1522688907.927:619): arch=x86_64 syscall=chmod
success=no
exit=EACCES a0=5633eefb7310 a1=1c0 a2=0 a3=7f3c78d1d6a0 items=0 ppid=
12116 pid=12142
auid=4294967295 uid=1001 gid=0 euid=1001 suid=1001 fsuid=1001 egid=0
sgid=0 fsgid=0
tty=(none) ses=4294967295 comm=initdb exe=/usr/lib/postgresql/9.4/bin/
initdb
subj=system_u:system_r:svirt_lxc_net_t:s0:c459,c1008 key=(null)

Hash: chown,svirt_lxc_net_t,usr_t,dir,setattr
```

This report shows the following suggested solution:

```
If you want to allow chown to have setattr access on the data directory
Then you need to change the label on data
Do
# semanage fcontext -a -t FILE_TYPE 'data'
where FILE_TYPE is one of the following: cgroup_t, nfs_t, svirt_home_t,
svirt_sandbox_file_t.
Then execute:
restorecon -v 'data'
```

Following is the command you would enter to implement this solution:

```
semanage fcontext -a -t svirt_sandbox_file_t "/opt/
postgresql-storage(/.*)?" restorecon -R /opt/postgresql-
storage
```

Note that the suggested solution was to apply the permissions only to the data folder for PostgreSQL. However, because you need write access for the ThingWorx tablespace, you must grant the same permissions to everything in that folder.

This example illustrates an issue with the ThingWorx platform. In this case, `sealert` produces the following alert:

```
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------
-------

SELinux is preventing /bin/mv from write access on the directory
ThingworxStorage.

*****  Plugin catchall_labels (83.8 confidence) suggests
******************

If you want to allow mv to have write access on the ThingworxStorage
directory
Then you need to change the label on ThingworxStorage
Do
# semanage fcontext -a -t FILE_TYPE 'ThingworxStorage'
where FILE_TYPE is one of the following: cgroup_t, container_var_lib_t,
nfs_t,
svirt_home_t, svirt_sandbox_file_t, tmpfs_t, virt_home_t.
Then execute:
restorecon -v 'ThingworxStorage'


*****  Plugin catchall (17.1 confidence) suggests
*************************

If you believe that mv should be allowed write access on the
ThingworxStorage
directory by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'mv' --raw | audit2allow -M my-mv
# semodule -i my-mv.pp
```

```
Additional Information:
Source Context                system_u:system_r:svirt_lxc_net_
t:s0:c320,c876
Target Context                unconfined_u:object_r:usr_t:s0
Target Objects                ThingworxStorage [ dir ]
Source                        mv
Source Path                   /bin/mv
Port                          <Unknown>
Host                          <Unknown>
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.13.1-166.el7_4.9.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     ip-10-99-0-109.ec2.internal
Platform                      Linux ip-10-99-0-109.ec2.internal
                              3.10.0-693.5.2.el7.x86_64 #1 SMP Fri Oct 20
                              20:32:50 UTC 2017 x86_64 x86_64
Alert Count                   13
First Seen                    2018-04-02 17:13:18 UTC
Last Seen                     2018-04-02 17:13:23 UTC
Local ID                      1759a007-f347-4d80-99e8-ee414eae7602

Raw Audit Messages
type=AVC msg=audit(1522689203.86:708): avc:  denied  { write } forpid=
14905
comm="java" name="ThingworxStorage" dev="nvme0n1p1" ino=864027929
scontext=system_u:system_r:svirt_lxc_net_t:s0:c320,c876
tcontext=unconfined_u:object_r:usr_t:s0 tclass=dir


type=SYSCALL msg=audit(1522689203.86:708): arch=x86_64 syscall=mkdir
success=no
exit=EACCES a0=7f6480f76e80 a1=1ff a2=7f6480f76e80 a3=b items=0 ppid=
14734
pid=14905 auid=4294967295 uid=1337 gid=1337 euid=1337 suid=1337 fsuid=
1337
egid=1337 sgid=1337 fsgid=1337 tty=(none) ses=4294967295 comm=java exe=/
opt/
jdk1.8.0_121/bin/java subj=system_u:system_r:svirt_lxc_net_t:s0:c320,c876
key=(null)

Hash: mv,svirt_lxc_net_t,usr_t,dir,write
```

This report shows the following suggested solution:

```
# semanage fcontext -a -t FILE_TYPE 'ThingworxStorage'
where FILE_TYPE is one of the following: cgroup_t, container_var_lib_t,
nfs_t,
svirt_home_t, svirt_sandbox_file_t, tmpfs_t, virt_home_t.
Then execute:
restorecon -v 'ThingworxStorage'
```

Following is the command you use to implement this solution:

```
semanage fcontext -a -t svirt_sandbox_file_t "/opt/thingworx-storage(/
.*)?"
         restorecon -R /opt/thingworx-storage
```

C

Using an External Microsoft SQL Server Database

The following example is a Docker Compose file for Microsoft SQL Server that uses an external database:

```
version: '2.2'

# Note, this example includes a _blank_ mssql docker image for demonstration
# purposes. The mssql service should be removed and DATABASE_HOST should point
# to the production database either with a DNS name, or IP.

services:
   mssql:
      image: microsoft/mssql-server-linux:2017-latest
      ports:
         - "1433"
      healthcheck:
         test: /opt/mssql-tools/bin/sqlcmd -U SA -P "$${SA_PASSWORD}" -h -1
                  -Q "set nocount on; select serverproperty('ServerName')" | grep -w "$${HOSTNAME}"
         interval: 15s
      environment:
         # NOTE: You must set SA_PASSWORD manually if using this image for testing.
         - "SA_PASSWORD="
         - "ACCEPT_EULA=y"

   platform:
      image: thingworx/platform-mssql:latest
      healthcheck:
         test: curl -s -w '%{http_code}' -U 'bad:creds'
```

```
            localhost:8080/Thingworx/Subsystems/PlatformSubsystem | grep -w 401
    interval: 15s
depends_on:
    mssql:
        condition: service_healthy
ports:
    - "8080:8080"
    - "8443:8443"
environment:
    - "INITIAL_HEAP=2"
    - "MAX_HEAP=4"
    - "DATABASE_HOST=mssql"
    - "DATABASE_PORT=1433"
    - "TWX_DATABASE_USERNAME=thingworx"
    - "TWX_DATABASE_SCHEMA=thingworx"
    # NOTE: TWX_DATABASE_PASSWORD for MSSQL platform must be set to match your
    # environment.
    - "TWX_DATABASE_PASSWORD="
    - "ENABLE_HTTP=true"
    - "ENABLE_HTTPS=false"
    # NOTE: If supplying a keystore for SSL, you must set your keystore password
    #- "SSL_KEYSTORE_PASSWORD="
    - "DATABASE_ADMIN_USERNAME=SA"
    # NOTE: DATABASE_ADMIN_PASSWORD should match your environment or SA_PASSWORD
    # if using the above image for testing.
    - "DATABASE_ADMIN_PASSWORD="
volumes:
    - "./thingworx-mssql-storage/ThingworxPlatform:/ThingworxPlatform"
    - "./thingworx-mssql-storage/ThingworxStorage:/ThingworxStorage"
    - "./thingworx-mssql-storage/ThingworxBackupStorage:/ThingworxBackupStorage"
    - "./thingworx-mssql-storage/tomcat-logs:/opt/apache-tomcat/logs"
```

D

Using an External PostgreSQL Database

The following example is a Docker Compose file for PostgreSQL that uses an external database:

```
version: '2.2'

# Note, this example includes a _blank_ postgreSQL docker image for demonstration
# purposes. The postgresql service should be removed and DATABASE_HOST should point
# to the production database either with a DNS name, or IP.

services:
  postgresql:
    image: postgres:9.4
    ports:
      - "5432"
    healthcheck:
      test: pg_isready -U postgres
      interval: 15s
    environment:
      # NOTE: You must manually set a default postgres password if using this
      # example for testing.
      - "POSTGRES_PASSWORD="
    volumes:
      - "./thingworx-postgres-storage/postgres-data:/var/lib/postgresql/data"

  platform:
    image: thingworx/platform-postgres:latest
    healthcheck:
```

```
    test: curl -s -w '%{http_code}' -U 'bad:creds'
          localhost:8080/Thingworx/Subsystems/PlatformSubsystem | grep -w 401
    interval: 15s
depends_on:
    postgresql:
      condition: service_healthy
ports:
    - "8080:8080"
    - "8443:8443"
environment:
    - "ENABLE_CLUSTERED_MODE=false"
    - "INITIAL_HEAP=2"
    - "MAX_HEAP=4"
    - "DATABASE_HOST=postgresql"
    - "DATABASE_PORT=5432"
    - "TWX_DATABASE_USERNAME=thingworx"
    - "TWX_DATABASE_SCHEMA=thingworx"
    # NOTE: TWX_DATABASE_PASSWORD must be set for your environment manually.
    - "TWX_DATABASE_PASSWORD="
    - "ENABLE_HTTP=true"
    - "ENABLE_HTTPS=false"
    # NOTE: If supplying a keystore for SSL, you must set your keystore password
    #- "SSL_KEYSTORE_PASSWORD="
    - "DATABASE_ADMIN_USERNAME=postgres"
    # NOTE: DATABASE_ADMIN_PASSWORD should match your environment or POSTGRES_PASSWORD
    # if using above service for testing.
    - "DATABASE_ADMIN_PASSWORD="
    - "TABLESPACE_LOCATION=/var/lib/postgresql/data"
volumes:
    - "./thingworx-postgres-storage/ThingworxPlatform:/ThingworxPlatform"
    - "./thingworx-postgres-storage/ThingworxStorage:/ThingworxStorage"
    - "./thingworx-postgres-storage/ThingworxBackupStorage:/ThingworxBackupStorage"
    - "./thingworx-postgres-storage/tomcat-logs:/opt/apache-tomcat/logs"
```