Dockerized midPoint

Introduction

Here we describe Docker image for midPoint.

The image can be found in the Evolveum/midpoint-docker GitHub project.

Besides the image itself, the project contains a couple of demonstrations of its usage:

Demonstration directory	Description
`demo/postgresql`	Demonstration of how to use an PostgreSQL repository.
`demo/extrepo`	Demonstration of how to use externally hosted repository. It also shows database schema version mismatch detection as well as automated upgrade procedure.
`demo/clustering`	This demo shows how to run two midPoint nodes working against common midPoint repository.
`demo/simple`	This demo shows Midpoint with PostgreSQL repository. Midpoint contains csv-resource as the source and openldap-resource as the target.

Prerequisites

In order to set up and run this container and associated demonstrations, you need a Linux machine with a reasonably recent Docker and `docker-compose` installation.

This container and demos were tested on Ubuntu 18.04.2 LTS with

- Docker 18.09.2,
- docker-compose 1.17.1.

Also, make sure appropriate ports are available on the host machine. They are listed in the documentation to individual demonstrations; usually they are 8080 and 5437, sometimes 8081, or 5432.

Getting started

The easiest way how to start dockerized midPoint is to use only image which use h2 database (Supported only in embedded mode. Not supported for production deployments. Only the version specifically bundled with midPoint is supported.). You have two choices: download image or build own image.

Download image

You can download image from docker hub.

```
$ docker pull evolveum/midpoint
```

Building own images

The above commands download evolveum/midpoint image from the Docker hub. Alternatively, you can build this image yourself. Here is how:

```
$ git clone https://github.com/Evolveum/midpoint-docker.git
$ cd midpoint-docker
$ ./build.sh
```

(Note the `build.sh` has a `-n` switch that skips downloading the midPoint distribution archive, saving some time during repeated builds or you can use custom midpoint-dist.tar.gz file.)

Starting

Running of image on port 8080:

```
$ docker run -p 8080:8080 --name midpoint evolveum/midpoint
```

If you use build from repository, then can start one of our demonstrations, e.g. postgresql.

```
$ cd demo/postgresql/
$ docker-compose up --build
```

After starting

After docker run or `docker-compose up` command successfully finishes you should see something like this on the console:

```
midpoint_server_1  | 2019-02-22 15:07:50,222 [] [main] INFO (org.springframework.boot.web.embedded.tomcat.
TomcatWebServer): Tomcat started on port(s): 8080 (http) with context path '/midpoint'
midpoint_server_1  | 2019-02-22 15:07:50,230 [] [main] INFO (com.evolveum.midpoint.web.boot.
MidPointSpringApplication): Started MidPointSpringApplication in 74.425 seconds (JVM running for 77.109)
```

Now you can log into midPoint using https://localhost:8080/midpoint URL, with an user of `administrator` and a password of `5ecr3t`.

Configuring the container (general information)

Before running of the image we can define some of the environment properties. In the case of composition the lowest level of configuration of the midPoint container is during its inclusion into a Docker composition. There is the full set of environment properties and other configurable items (e.g. Docker secrets and configs) available.

During the composition some of the environment properties can be made accessible from the outside. This depends strictly on the compositor. The demonstrations here show some of the options.

How to set environment variables before running of the image

You can set the environment variables like this:

```
$ docker run -p 8080:8080 -e MP_MEM_MAX="4096M" -e MP_MEM_INIT="4096M" --name midpoint evolveum/midpoint
```

How to set environment variables after composition is done

After the composition is done, you can set the environment variables like this:

```
$ export MP_MEM_MAX="4096M" MP_MEM_INIT="4096M"
$ docker-compose up --build
```

Or like this:

```
$ env MP_MEM_MAX="4096M" MP_MEM_INIT="4096M" docker-compose up --build
```

How to set Docker secrets and configs

The way of accessing secrets and configs is specific to the composition. In our demonstrations these are stored in the `configs-and-secrets` directory. They are provided to midPoint containers in appropriate ways. (Currently, secrets are passed as Docker secrets, configs are mounted as volumes. This might be changed in the future.) For detailed information on individual items please see the following sections.

Configuring specific container features

In this section we describe how to configure and use specific features of this midPoint dockerization.

Repository

Repository configuration is done via the following environment variables.

Environment variable	Meaning	Default value
`REPO_DATABASE_TYPE`	Type of the database. Supported values are `mariadb`, `mysql`, `postgresql`, `sqlserver`, `oracle`. It is possible to use `H2`as well but H2 is inappropriate for production use.	`h2`

`REPO_JDBC_URL`	URL of the database.	`H2: jdbc:h2:tcp://$REPO_HOST:$REPO_PORT/$REPO_DATABASE`
		
		`MariaDB: jdbc:mariadb://$REPO_HOST:$REPO_PORT/$REPO_DATABASE?characterEncoding=utf8`
		
		MySQL: `jdbc:mysql://$REPO_HOST:$REPO_PORT/$REPO_DATABASE?characterEncoding=utf8`
		
		PostgreSQL: `jdbc:postgresql://$REPO_HOST:$REPO_PORT/$REPO_DATABASE`
		
		SQL Server: `jdbc:sqlserver://$REPO_HOST:$REPO_PORT;database=$REPO_DATABASE`
		
		Oracle: `jdbc:oracle:thin:@$REPO_HOST:$REPO_PORT/xe`

`REPO_HOST`	Host of the database. Used to construct the URL.	`midpoint_data`
`REPO_PORT`	Port of the database. Used to construct the URL.	`5437, 3306, 5432, 1433, 1521 for H2, MariaDB/MySQL, PostgreSQL, SQL Server and Oracle, respectively`

REPO_DATABASE	Specific database to connect to. Used to construct the URL.	midpoint
REPO_USER	User under which the connection to the database is made.	midpoint
REPO_PASSWORD_FILE	File (e.g. holding a docker secret) that contains the password for the db connection.	
REPO_MISSING_SCHEMA_ACTION	What should midPoint do if the database schema is missing (options: warn, stop, create).	create
REPO_UPGRADEABLE_SCHEMA_ACTION	What should midPoint do if the database schema is obsolete but upgradeable (options: warn, stop, upgrade). As of midPoint 3.9, the only automated transition available is from 3.8 to 3.9.	stop
REPO_SCHEMA_VERSION_IF_MISSING	For midPoint versions before 3.9 that do not have schema information explicitly stored in the database, this parameter allows specifying the version externally. It can be used for automated upgrade from 3.8 to 3.9. (In such cases, specify it to be 3.8, assuming this is your schema version.)	
REPO_SCHEMA_VARIANT	Used to specify what schema variant is to be used for automated creation or upgrade of the database schema. Currently the only known variant is utf8mb4 for MySQL/MariaDB. **Beware:** it is the administrator's responsibility to choose the correct variant! Currently midPoint does not try to determine the variant present in the database. So be sure to avoid applying e.g. mysql-upgrade-3.8-3.9-utf8mb4.sql if the database is not in utf8mb4 character set, or vice versa.	

For automatic schema creation and upgrade options please see Schema creation and updating section in midPoint documentation.

Note that in order to connect to the database you have to provide the password. For security reasons, we use the indirect way through file access. So, typically you provide the following Docker secret:

Secret	Meaning	Typical location in demonstration scenarios
mp_database_password.txt	A password used to access the repository (relates to REPO_USER).	configs-and-secrets/midpoint/database_password.txt

Of course, you can provide the password file in any other way, assuming you correctly set REPO_PASSWORD_FILE environment variable.

Other

Other aspects can be configured using the following variables and Docker secrets or configs.

Environment variable	Meaning	Default value
MP_MEM_MAX	The limit for Java heap memory (-Xmx setting)	2048m
MP_MEM_INIT	The initial amount of Java heap memory (-Xms setting)	1024m
MP_JAVA_OPTS	Any other Java options to be passed to midPoint	
MP_KEYSTORE_PASSWORD_FILE	File (e.g. holding a docker secret) that contains the password for the midPoint keystore	
MP_DIR	midPoint home directory. Do not change until absolutely necessary, as the change might break many things.	/opt/midpoint
MP_DIST_FILE	Name of midpoint-dist file from which will be started Midpoint. It must have .tar.gz type and it's path must be {path_to_repository}/midpoint-docker/MP_DIST_FILE. This file will be download during building of the image, but you can use custom midpoint-dist file with *./build.sh -n*.	midpoint-dist.tar.gz
TIMEZONE	Name of the time zone to be set for the container upon startup. E.g. US/Central.	UTC

And the following Docker secrets are to be provided:

Item	Kind	Meaning	Location
mp_keystore_password.txt	secret	Java keystore password used by midPoint e.g. to encrypt sensitive information stored in the repository.	configs-and-secrets/midpoint/keystore_password.txt

`jmxremote.password`
 	
`jmxremote.access`
	secret	Names of the password and access files for JMX authentication and authorization. Use for clustering. For more information see Clustering / high availability setup. These secrets are specific for clustering demo.	`configs-and-secrets/midpoint/jmxremote.password`
			
			configs-and-secrets/midpoint/jmxremote.access