Getting Started with midPoint



This is a live document. If you encounter issues, please let us know via the `#incommon-midpoint` Internet2 Slack channel or via email to inctrust-si@incommon.org

- Introduction
- Prerequisites
- Getting started (simple demo)
  - Building own images
  - After starting
- Beyond the simple demo
- Configuring the container (general information)
  - How to set environment variables after composition is done
  - How to set Docker secrets and configs
- Configuring specific container features
  - Repository
  - Logging
  - Authentication
  - Other
    - Guiding requirements for this project: TIER Docker Container Specification

Introduction

This page shows how to get started with a Docker image for the midPoint component of the InCommon Trusted Access Platform.

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. The most advanced `demo/grouper` optionally uses an LDAP browser, e.g. Apache Directory Studio that itself requires Java.

This container and demos were tested on Ubuntu 18.04.1 LTS with

- Docker 18.06.1-ce,
- docker-compose 1.17.1 (and 1.22.0),
- libxml2-utils package (in order to have `xmllint` command available),
- Apache Directory Studio 2.0.0.v20170904-M13 with OpenJDK 8.

The LDAP browser is really optional. It is used only to check that LDAP objects are created correctly. You can safely proceed without it.

It seems that the usual way of installation via e.g. `apt-get` does not always guarantee sufficiently recent versions of Docker and `docker-compose`. To have the latest stable versions you can use procedures described here:

- https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-docker-ce
- https://docs.docker.com/compose/install/

Also, make sure appropriate ports are available on the host machine. They are listed in the documentation to individual demonstrations; usually they are 8443 and 3306, sometimes 389, 443, or 5432. The `demo/grouper` needs even more free ports, please see the description.

Getting started (simple demo)

The image cannot be "run" by itself as it requires a midPoint repository - i.e. an SQL database - to execute. So the easiest way to start dockerized midPoint is to use one of the provided demonstrations. The most logical choice for just getting started with midPoint is `demo/simple`.

```
$ git clone https://github.internet2.edu/Docker/midPoint_container.git
$ cd midPoint_container/demo/simple
$ docker-compose up
```

Building own images

The above commands download TAP `midpoint` image from Internet2's enterprise github. Alternatively, you can build this image yourself. Here is how:

```
$ git clone https://github.internet2.edu/Docker/midPoint_container.git
$ cd midPoint_container
$ ./build.sh
$ cd demo/simple
$ docker-compose up
```

(Note the `build.sh` has a `-n` switch that skips downloading the midPoint distribution archive, saving some time during repeated builds.)

After starting

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

```
midpoint_server_1  | midpoint;midpoint.log;demo;;2018-09-20 16:25:22,191 [] [main] INFO (org.springframework.
boot.web.embedded.tomcat.TomcatWebServer): Tomcat started on port(s): 8080 (http) 9090 (http) with context path
'/midpoint'
midpoint_server_1  | midpoint;midpoint.log;demo;;2018-09-20 16:25:22,209 [] [main] INFO (com.evolveum.midpoint.
web.boot.MidPointSpringApplication): Started MidPointSpringApplication in 60.512 seconds (JVM running for
61.688)
```

Now you can log into midPoint using https://localhost:8443/midpoint URL, with an user of `administrator` and a password of `5ecr3t`. If you are using midPoint 4.8.1 or greater please check the documentation on how to set an initial password.

Beyond the simple demo

Besides the midPoint image itself, and the simple demo discussed above, the project contains demonstrations that integrate midPoint with selected other components

Demonstration Instruction Pages	Description
`demo/simple`	The simplest use of midPoint: just running it along with a dockerized MariaDB repository.
`demo/shibboleth`	Shows how to use midPoint with the Shibboleth authentication.
`demo/postgresql`	Demonstration of how to use an alternative repository (PostgreSQL running in a Docker container) instead of MariaDB-based one.
`demo/extrepo`	Demonstration of how to use externally hosted repository instead of MariaDB-based one. It also shows database schema version mismatch detection as well as automated upgrade procedure.
`demo/grouper`	This is a demonstration of the use of midPoint image in a wider environment consisting of Grouper, Shibboleth, LDAP directory, RabbitMQ messaging, and sample source and target systems. For a step-by-step walk-through, see the Grouper Integration Demo page.

Configuring the container (general information)

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 after composition is done

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

```
$ export ENV="test" USERTOKEN="4.0.1" MP_MEM="4096m"
$ docker-compose up
```

Or like this:

```
$ env ENV="test" USERTOKEN="4.0.1" MP_MEM="4096m" docker-compose up
```

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.	`mariadb`
`REPO_JDBC_URL`	URL of the 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.	3306, 5432, 1433, 1521 for MariaDB/MySQL, PostgreSQL, SQL Server and Oracle, respectively
`REPO_DATABASE`	Specific database to connect to. Used to construct the URL.	`registry`
`REPO_USER`	User under which the connection to the database is made.	`registry_user`
`REPO_PASSWORD_FILE`	File (e.g. holding a docker secret) that contains the password for the db connection.	`/run/secrets/mp_database_password.txt`
`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 4.0, 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/application/database_password.txt

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

Logging

Logging is configured by setting the following environment variables:

Environment variable	Meaning	Default value
ENV	environment (e.g. prod, dev, test)	demo
USERTOKEN	arbitrary user-supplied token	

According to the specification, semicolons and spaces in these fields are eliminated. We decided to replace them by underscores.

Authentication

This midPoint dockerization supports two authentication mechanisms.

Mechanism	Description
internal	Users are authenticated against midPoint repository. Login name to be used is the name property of the user, and the password is credentials/password/value property.
shibboleth	Users are authenticated against Shibboleth IdP. This is ensured using Shibboleth SP (service provider) module for Apache httpd configured as reverse proxy for midPoint.

Authentication configuration is done using the following environment variables.

Environment variable	Meaning	Default value
AUTHENTICATION	Authentication mechanism to use	internal
LOGOUT_URL	URL to be used for logout (used for Shibboleth authentication)	https://localhost:8443/Shibboleth.sso/Logout
SSO_HEADER	Shibboleth attribute to be used as a login identifier. It is matched against name property of the user when logging in. When changing it, do not forget to change your Shibboleth IdP configuration as well as midPoint's shibboleth2.xml configuration file.	uid

Note that besides these variables you have to provide the following files. They are necessary for the Shibboleth service provider module.

File	Description	Typical location in demonstration scenarios
/etc/shibboleth/idp-metadata.xml	Metadata related to Shibboleth identity provider	configs-and-secrets/midpoint/shibboleth/idp-metadata.xml
/etc/shibboleth/shibboleth2.xml	Service provider configuration	configs-and-secrets/midpoint/shibboleth/shibboleth2.xml

`/etc/shibboleth/sp-cert.pem`	Service provider certificates file	`configs-and-secrets/midpoint/shibboleth/sp-cert.pem`

And the following Docker secrets are to be provided:

Secret	Description	Typical location in demonstration scenarios
`mp_sp-key.pem`	Service provider private key	`configs-and-secrets/midpoint/shibboleth/sp-key.pem`

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	`/run/secrets/mp_keystore_password.txt`
`MP_DIR`	midPoint home directory. Do not change until absolutely necessary, as the change might break many things.	`/opt/midpoint`
`TIMEZONE`	Name of the time zone to be set for the container upon startup. E.g. `US/Central`.	`UTC`

Other files that are necessary for this midPoint container to function are:

Item	Meaning	Location
`/etc/pki/tls/certs/host-cert.pem`	Host certificate for Apache httpd	`configs-and-secrets/midpoint/httpd/host-cert.pem`
`/etc/pki/tls/certs/cachain.pem`	Certificate chain for Apache httpd	`configs-and-secrets/midpoint/httpd/host-cert.pem`

And the following Docker secrets are to be provided:

Item	Kind	Meaning	Location
`mp_host-key.pem`	secret	Private key for Apache httpd	`configs-and-secrets/midpoint/httpd/host-key.pem`
`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/application/keystore_password.txt`

Guiding requirements for this project: TIER Docker Container Specification