Swagger Editor 3.0 Docker Container

Introduction

The intent of this document is to provide a set of steps that a reader can use to create a self-contained Docker container for API-First development using latest Swagger Editor (3.x) and OpenAPI (2 or 3).
A Dockerfile is provided to short-circuit the process.

The container will have the means to:

- Run the Swagger Editor Server
- Convert YAML specification documents to JSON and the vice versa

The container is based on the latest Docker node image with extras.

The container uses:

- Swagger Editor Distributable (`swagger-editor-dist`)

- `swagger-codegen-cli/3.0.20` to support YAML to JSON conversion and generation of client and server stubs based on the OpenAPI Specification / Swagger file for supported languages. `swagger-codegen-cli` requires Java 8, which is installed during container setup.
- `nodemon` server
- `http-server` server

[Top]

Assumptions

It is assumed in the text that Windows 10 with Debian WSL is the Host operating environment.

Make such changes, to the small number of commands that this affects, as you need to make it work in a regular Linux environment.

[Top]

Instant Gratification Docker Container

In this section we will create a docker container with all that can be pre-installed and pre-configured to run the Swagger Editor server and convert between YAML and JSON openapi specifications.

It is assumed that the Host build environment is Windows 10 with Debian WSL (Windows Subsystem for Linux), Docker installed in Windows 10 and all Host work done using WSL Debian bash shell.

> Once the image is built it will not matter in what environment it was built and docker commands to create and manage the container are much the same regardless of the docker host environment.

If this does not match your environment then you will need to make such adjustments as might be required. At most Host paths are likely to require changes from something like `/mnt/d/...` to something like `D:/...` or `/usr/home/myself/...` or `/opt/dockerwork/...` or some such.

[Top]

Host Artefacts

Create Host directories

Adjust Host paths above directory named `api` as you see fit.

```
HOST_DIR=/mnt/d/github_materials

mkdir -pv ${HOST_DIR}/swagger_editor/api
cd ${HOST_DIR}/swagger_editor
```

[Top]

Create the example openapi.yaml API Specification (OpenAPI 3.0.1)

We need example openapi specifications in yaml and json in the Docker image.

The easiest way to example specs to the Image is to create them on the host and copy them to the image while building it.

```
HOST_DIR=/mnt/d/github_materials

cat <<-'EOF' > ${HOST_DIR}/swagger_editor/api/openapi.yaml
openapi: "3.0.1"
info:
  title: Weather API
  description: |
    This API is a __test__ API for validation of local swagger editor
    and swagger ui deployment and configuration
  version: 1.0.0
servers:
  - url: 'http://localhost:3003/'
tags:
  - name: Weather
    description: Weather, and so on
paths:
  /weather:
    get:
      tags:
        - Weather
      description: |
        It is __Good__ to be a _King_
        And a *Queen*
        And a _Prince_
        And a __Princess__
        And all the Grandchildren
        And their Children
      operationId: getWeather
      responses:
        '200':
          description: 'All is _well_, but not quite'
          content: {}
        '500':
          description: Unexpected Error
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/response_500'
components:
  schemas:
    response_500:
      type: object
      properties:
        message:
          type: string
```

```
EOF
```

[Top]

Create Docker Image

The Dockerfile and following instructions will create a baseline Docker Image. The Image will be able to be used for spinning up Container instances as needed with minimum extra work required to make them useable for API development work on different APIs.

[Top]

Create Dockerfile in the Host directory

Change timezone and exposed ports as required.

```
HOST_DIR=/mnt/d/github_materials

cat <<-'EOF' > ${HOST_DIR}/swagger_editor/Dockerfile
FROM node:latest

ENV TZ_PATH="Australia/Sydney"
ENV TZ_NAME="Australia/Sydney"
ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y \
    dos2unix \
    openjdk-8-jdk && \
  \
  npm i -g \
    http-server \
    nodemon && \
    \
  # set timezone
  #
  cp -v /usr/share/zoneinfo/${TZ_PATH} /etc/localtime && \
  echo "${TZ_NAME}" > /etc/timezone && \
  \
  # create the api directory (to be masked by bound volume if required)
  #
  mkdir -pv /api

COPY api/openapi.yaml /api

RUN \
\
  # copy openapi.yaml to swagger-editor
```

```
  #
  mkdir -pv /swagger_tools/swagger-editor && \
  cp -v /api/openapi.yaml /swagger_tools/swagger-editor && \
\
  # "install" swagger-codegen
  #
  mkdir -pv /swagger_tools/swagger-codegen && \
  wget https://repo1.maven.org/maven2/io/swagger/codegen/v3/swagger-codegen-
cli/3.0.20/swagger-codegen-cli-3.0.20.jar -O /swagger_tools/swagger-
codegen/swagger-codegen-cli.jar && \
\
  # convert yaml to jason and back again
  #
  cd /swagger_tools/swagger-editor/ && \
  java -jar /swagger_tools/swagger-codegen/swagger-codegen-cli.jar generate -i
/swagger_tools/swagger-editor/openapi.yaml -l openapi -o /swagger_tools/swagger-
editor && \
  java -jar /swagger_tools/swagger-codegen/swagger-codegen-cli.jar generate -i
/swagger_tools/swagger-editor/openapi.json -l openapi-yaml -o
/swagger_tools/swagger-editor/converted && \
\
  # Install swagger-editor
  #
  cd /swagger_tools/  && \
  npm init -y && \
  npm i swagger-editor-dist  && \
  cp -rv node_modules/swagger-editor-dist/* /swagger_tools/swagger-editor && \
\
  # update index.html to use the local openapi.json
  #
  cd /swagger_tools/swagger-editor && \
  sed -i "s|const editor = SwaggerEditorBundle({|const editor =
SwaggerEditorBundle({ url: 'http://localhost:3001/openapi.yaml',|" index.html && \
  echo "Done" && \
\
  # create "run editor" server script
  #
  echo '#!/bin/bash' > /swagger_tools/run_editor_server.sh && \
  echo 'cd /swagger_tools/swagger-editor' >> /swagger_tools/run_editor_server.sh
&& \
  echo 'nodemon -L -w index.html -w /api/openapi.yaml -w /api/*.pdf -x "cp -v
/api/* /swagger_tools/swagger-editor/ && http-server -p 3001"' >> ${0}.log >>
/swagger_tools/run_editor_server.sh && \
  chmod u+x /swagger_tools/run_editor_server.sh && \
\
  sed -i '/set -e/a test $( ps -C run_editor_serv -o stat --no-headers ) == "S" ||
nohup /swagger_tools/run_editor_server.sh </dev/null &' /usr/local/bin/docker-
entrypoint.sh

EXPOSE 3001

EOF
```

[Top]

Create baseline Docker Image

The following command will create the baseline image with specific packages pre-installed and the timezone set.

```
CONTAINER_NAME=swagger_editor
IMAGE_VERSION=1.0.0

docker build \
    --tag ${CONTAINER_NAME}:${IMAGE_VERSION} \
    --force-rm .
```

At this point we have the new image `swagger_editor:1.0.0`ready to roll.

From this point, until the image is deleted, we can spin up a container, based on this image, in a matter of seconds.

[Top]

Start the Docker Container Instance

Create a container based on the new Image

Now that we have the baseline image we can create and explore a container that uses it.

```
IMAGE_VERSION="1.0.0"
IMAGE_NAME="swagger_editor"
CONTAINER_NAME="swagger_editor"
CONTAINER_HOSTNAME="swagger_editor"
CONTAINER_MAPPED_PORTS=" -p 127.0.0.1:3001:3001/tcp "

docker run \
    --name ${CONTAINER_NAME} \
    --hostname ${CONTAINER_HOSTNAME} \
    ${CONTAINER_MAPPED_PORTS} \
    --detach \
    --interactive \
    --tty\
        ${IMAGE_NAME}:${IMAGE_VERSION}
```

[Top]

Connect to the running container

The following command will connect us to teh running container and offer us the interactive shell to work in:

```
docker exec -it -w='/api' swagger_editor bash -l
```

[Top]

Test Swagger Editor

The swagger-editor server is started when the container is created and started, and re-starts when the container is restarted.

With the container running, in a host web browser open the pre-configured API specification in the swagger-editor.

http://localhost:3001/#

Please note that the Swagger Editor is running in the Host's Web Browser and that it can import OpenAPI specifications from the Host and export / save OpenAPI specifications ot the host, this it can be used as a local Swagger Editor on the Host.

Important Notes

The Swagger Editor server's `index.html` has been rigged to serve the file `openapi.yaml` from its own directory.

The command in the Dockerfile is reproduced below.

```
nodemon -L -w index.html -w /api -x "cp -v /api/* /swagger_tools/swagger-editor/
&& http-server -p 3001"
```

It instructs `nodemon` to watch for changes to files in directory `/api`.
If a change is detected, `nodemon` will copy all files from that directory to the `/swagger_tools/swagger-editor` directory and will restart the `http-server`.

Please note that the Swagger Editor runs in the Web Browser in the Host environment.
Unless the container startup command is configured so that it mounts a host directory over the top of the containers `/api` directory, Swagger Editor will have no ability to write to the `/api` directory in the container, so any changes through the Swagger Editor will be local to the Host.

As given in the example, as soon as the `openapi.yaml` changes in the `api` directory, the server will be restarted and the Web Browser on the Host can be refreshed to load the changed version of `openapi.yaml`.

[Top]

Must I edit openapi.yaml file in the container?

No.

As mentioned at the last paragraph in section Test Swagger Editor, this container can be used as a local Swagger Editor for editing OpenAPI files in Host directories.

How to edit the yaml file in the Container

There are at least 3 different ways in which the container's `/api/openapi.yaml` file can be edited such that changes persist across container re-creation.

Use docker cp command

Docker has the ability to copy files from the Host to the Container and vice versa.

Here is the `docker cp` usage:

```
"docker cp" requires exactly 2 arguments.
See 'docker cp --help'.

Usage:  docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
        docker cp [OPTIONS] SRC_PATH|- CONTAINER:DEST_PATH

Copy files/folders between a container and the local filesystem
```

Assuming the container as discussed so far, with `openapi.yaml` in the container directory `/api` being the ***source-of-truth***, here are the steps:

1. Copy `openapi.yaml` from the container to the Host
    1. Open a terminal window on the Host in the Host directory to which you want to copy the container's openapi.yaml
    2. execute `docker cp swagger_editor:/api/openapi.yaml ./`
2. Copy `openapi.yaml` from the Host to the Container
    1. Open a terminal window on Host in the Host directory to where oyu have the openapi.yaml file which you want to copy the container
    2. execute `docker cp ./openapi.yaml swagger_editor:/api`

Using this method one can edit the original `openapi.yaml` file in the local Swagger Editor, save it to the Host, and copy it to the container for the next Swagger Editor session.

Use VSCode Remote

If you use VSCode for development, and VSCode has the Remote Containers extension installed, you can connect to the container and use VSCode to edit files directly in the container.

Here are the steps:

1. Start the container as discussed so far.
2. Connect to the container as discussed in Connect to the running container
3. Start VSCode on the ***Host***
4. Click on the green '**><**' rectangle in the bottom-left corner (Shows "`Open Remote Window`" legend if one hovers the mouse over it)

5. From the dropdown, top-centre of the VSCode window, choose "`Remote-Containers: Attach to Running Container...`"
6. From the dropdown, select "`/swagger_editor: swagger_editor:1.0.0 ...`"
7. When the new VSCode window opens, click on the "`Open Folder`" button and enter `/api`
8. Edit the `openapi.yaml` file to your heart's content
9. Save
10. Refresh the Host Web Browser window to see changes

The remote VSCode might need OpenAPI / Swagger extensions for pretty-printing, snippets and so on. Pick any you like.

You can copy the `openapi.yaml` file between the Host and the Guest, in either direction, using docker cp command. See Use docker cp command.

Use bound volume

It is possible to share a Host directory with the container in such a way that changes made in one environment are visible in the other.

To effect this, one needs to start the docker container with a modified command line so that docker gets told what Host directory to share and where to "mount" it in the container's file system.

Steps:

1. On the Host, stop and remove the container if it is running: `docker container stop swagger_editor; docker container rm swagger_editor`
2. On the host, start the container with the following command, assuming `/mnt/d/github_materials/swagger_editor/api` is the host directory to share:

```
SOURCE_HOST_DIR=d:/github_materials/swagger_editor

IMAGE_VERSION="1.0.0"
IMAGE_NAME="swagger_editor"
CONTAINER_NAME="swagger_editor"
CONTAINER_HOSTNAME="swagger_editor"
CONTAINER_VOLUME_MAPPING=" -v ${SOURCE_HOST_DIR}/api:/api"
CONTAINER_MAPPED_PORTS=" -p 127.0.0.1:3001:3001/tcp "

docker run \
    --name ${CONTAINER_NAME} \
    --hostname ${CONTAINER_HOSTNAME} \
    ${CONTAINER_VOLUME_MAPPING} \
    ${CONTAINER_MAPPED_PORTS} \
    --detach \
    --interactive \
    --tty\
        ${IMAGE_NAME}:${IMAGE_VERSION}
```

4. Connect to the container as discussed in Connect to the running container

5. Use the Swagger Editor in the Host Web Browser to access and change the `openapi.yaml` file
6. When done, pull down the `File->Save as YAML` and save the file as `openapi.yaml` in the Host directory `d\:github_materials\swagger_editor\api`
7. In Guest, note that the `http-server` was re-started because `nodemon` recognised that the file in the directory it is watching changed.
8. In the container view the file `/api/openapi.yaml` and see the changes

Because the host and the container share the (bound) volume where the API specification exists, it is possible to edit the file on Host and in the container and see the changes in either environment.

[Top]

Other Notes

Initial API Specification Example YAML

Our original OpenAPI Specification was copied to the container file `/api/openapi.yaml` during Docker Image build.

[Top]

JSON Versions of Specification

Our subsequent manipulations, during Docker Image build, resulted in the creation of the following equivalents:

```
/swagger_tools/swagger-editor/openapi.json
/swagger_tools/swagger-editor/converted/openapi.yaml
```

[Top]

Where to change where Swagger Editor looks for the YAML specification and under what name?

`/swagger_tools/swagger-editor/index.html`. Please note that this change will persist once the docker container in which it is made is removed and re-created, unless this change is made in the Dockerfile and the image is re-built. Subsequently created containers that use the modified image will "inherit" the change.

[Top]

What port does Swagger Editor listen on?

Swagger Editor server inside the container listens on port 3001. To change the port on which the Host listens, change the port mapping the container start command uses.

For example:

```
CONTAINER_MAPPED_PORTS=" -p 127.0.0.1:3210:3001/tcp "
```

will change the port the Hosts maps to the container's 3001 from 3001 to 3210. The Host's web browser will need to use the url `http://localost:3210/#` to connect to the Swagger Editor served from container.

If you change the listening port make sure to adjust your `docker run ...` and `docker exec ...` commands otherwise you will not be able to connect to the listener in the container from outside.

[Top]

Next Steps

Now that one can edit the API using a Swagger Editor hosted in the local environment in a docker container it might be good to be able to generate API stubs to test the API.

This is coming in the next installment.

[Top]

Licensing

[Top]

2020/07 MCz