SBSGUI: Development Image

This image supports the use of a Docker container for the development of SBSGUI in an Ubuntu environment.

Table of Contents

1. Installed core components
2. Creating a new SBSGUI development container
3. Working with an existing SBSGUI development container

1. Installed core components

With the following command you can check in detail which software components in which versions are included in the Docker image:

```
apt list --installed
```

Version 1.0.0

Component	Version	Remark	Status
asdf	v0.8.1-95f2cdf	base version	
curl	7.68.0	base version	
dos2unix	7.4.0	base version	
Erlang/OTP	24.0.5		
G++ & GCC	9.3.0	base version	
Git	2.25.1	base version	
GNU Autoconf	2.69	base version	
GNU Automake	1.16.1	base version	
GNU make	4.2.1	base version	
htop	3.0.5		
Java	11.0.11	base version [openjdk]	
LCOV	1.14	base version	
Node.js [npm]	v14.17.5 [6.14.14]		
ODBC	2.3.6	base version	
OpenSSL	1.1.1k		

Component	Version	Remark	Status
Python3	3.8.10	base version	
rebar3	3.16.1		
tmux	3.2a		
Ubuntu	20.04.3 LTS	base version [focal]	
Vim	8.2.2269	base version	
wget	1.20.3	base version	
Yarn	n/a	asdf plugin is faulty	

2. Creating a new SBSGUI development container

2.1 Getting started

```
> REM Assumptions:
> REM   - you want to map the container port 8443 to the host port 443
> REM   - the name of the Docker container should be: my_sbsgui_dev
> REM   - the path the host repository is: //C/projects/my_repro
> REM   - the directory name for this repository inside the container should be:
my_repro_dir
> REM   - you want to use the latest version of the SBSGUI development image
> docker run -it -p 443:8443 \
             --name my_sbsgui_dev \
             -v //C/projects/my_repro:/my_repro_dir \
             konnexionsgmbh/sbsgui_dev:latest

> REM Stopping the container
> docker stop my_sbsgui_dev

> REM Restarting the container
> docker start my_sbsgui_dev

> REM Entering a running container
> docker exec -it my_sbsgui_dev bash
```

2.2 Detailed syntax

A new container can be created with the `docker run` command.

Syntax:

```
docker run -it
           [-p <port>:8443] \
```

```
            [--name <container_name>] \
            [-v <directory_repository>:/sbsgui] \
            konnexionsgmbh/sbsgui_dev[:<version>]
            [<cmd>]
```

Parameters:

- **port** - an optional listener port
- **container_name** - an optional container identification
- **directory_repository** - an optional host repository directory - the default value is expecting the repository inside the container
- **version** - an optional version number of the image or the constant `latest`
- **cmd** - an optional command to be executed in the container, default is `bash` for running the `bash` shell

Detailed documentation for the command `docker run` can be found here.

Examples:

1. Creating a new Docker container named `my_sbsgui_dev` using a repository inside the Docker container:

   ```
   docker run -it --name my_sbsgui_dev konnexionsgmbh/sbsgui_dev:latest
   ```

2. Creating a new Docker container named `my_sbsgui_dev` using the host repository of a Windows directory `D:\projects\sbsgui`:

   ```
   docker run -it --name sbsgui_dev -v //D/projects/sbsgui:/sbsgui
   konnexionsgmbh/sbsgui_dev:latest
   ```

3. Creating a new Docker container named `my_sbsgui_dev` using the host repository of a Linux directory `/sbsgui` and mapping port `8443` to port `8000`:

   ```
   docker run -it --name my_sbsgui_dev -p 8000:8443 -v /sbsgui:/sbsgui
   konnexionsgmbh/sbsgui_dev:latest
   ```

---

3 Working with an existing SBSGUI development container

3.1 Starting a stopped container

A previously stopped container can be started with the `docker start` command.

Syntax:

```
docker start <container_name>
```

Parameter:

- **container_name** - the mandatory container identification, that is an UUID long identifier, an UUID short identifier or a previously given name

Detailed documentation for the command `docker start` can be found here.

3.2 Entering a running container

A running container can be entered with the `docker exec` command.

Syntax:

```
docker exec -it <container_name> <cmd>
```

Parameter:

- **container_name** - the mandatory container identification, that is an UUID long identifier, an UUID short identifier or a previously given name
- **cmd** - the command to be executed in the container, e.g. `bash` for running the `bash` shell

Detailed documentation for the command `docker exec` can be found here.