

Using Docker Images

HOW THIS TUTORIAL WORKS

This tutorial is written in order to give a small insight in using Docker image files with Singularity. It will cover, connecting to cluster, pulling the image and starting a database for a small test script.

It is written as shell script itself and might just be executed on the cluster!

You may also just copy paste the commands for a better understanding.

Please check [this tutorial](#) if you are interested in creating an own image file.

Thx for reading

Jan Eberhardt

```
#!/bin/bash
# 0. Login on Frontend (you probably already did that)
# Use your TUB account and host gateway.hpc.tu-berlin.de
# ssh "<TUB account name>@gateway.hpc.tu-berlin.de"

# 1. Get Docker Image
# Go to your home directory and download the image via singularity.
# You must load the singularity module beforehand.
module load singularity/3.1.0

# Pulling docker images is done by Singularity's pull command. Source will
# be something like "docker://[package name]".
# Singularity will automatically download the latest version of the image
# and rewrite it to a Singularity image file (sif) as "[package
# name]_latest.sif".
# Therefore you will need write permission in your current working directory
# (which is why we changed into home).
cd
singularity pull "docker://mongo"

# 2. Create Python environment [if using Python]
# Most Python projects will use open source libraries installed by pip.
# Since normal users are not allowed to do so, it is recommended to install
# pip packages in user space or in a virtual python environment. We would
# discourage you from using user space for installation since most packages
# you will only use once in your life and it is therefore cleaner to get an
# unique environment for each project of yours.

# a) Load python module.
module load python/3.7.1

# b) Create the environment
py="$~/mongodb_venv"
```

[illegible]

```
"${py}/bin/pip3" install --upgrade pip
"${py}/bin/pip3" install -r "${py}/pip-packages"

# 3. Create DB directory
dd=~/.mongo"
mkdir -p "${dd}"

# 4. Start Server and run
# Use mongod_start.sbatch in order to allocate resources for and to start
mongod server:
# This script will open up a server on a node and close it after the Python
script finishes.
#
# We use the --exclusive switch of SBATCH in order to secure that port 27017
(default mongod) is not in use.
# If you do not like to use an exclusive node you will have to either accept
the risk that the command fails or
# to build a Singularity image of your own.
#>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>
cat << EOF > ./mongod_start.sbatch
#!/bin/bash
#
# Start MongoDB docker image
#
#SBATCH --job-name=MongoDBStart
#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --cpus-per-task=4
#SBATCH --exclusive
#
#1 prepare
module load singularity/3.1.0
#- start instance (not the server)
#- In that way we can use the instance command to stop the database when
script finishes.
singularity instance start --bind "${dd}:/data/db" ./mongo_latest.sif
mongod
#- start server (by runscrip)
#- It will generate a lot of output, better redirecting that to oblivion
(1>/dev/null).
#- Also this call will lock your shell, avoided by ending the command with
"&".
singularity run instance://mongod 1>/dev/null &

#2 run program
#- wait for database server to run
sleep 5
#- run script
${py}/bin/python3 ~./mongod run.py
```

