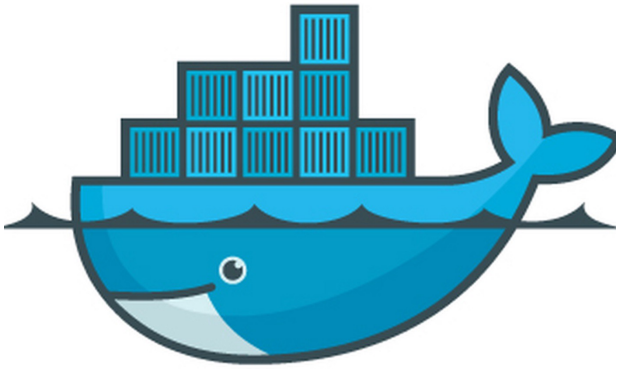
How to run tools for Remote Sensing and GIS data processing using Docker and Singularity?



Pete Bunting
Aberystwyth University
Earth Observation and Ecosystem Dynamics Group
Department of Geography and Earth Sciences
pfb@aber.ac.uk

ACSEM

PRIFYSGOL ABERYSTWYTH UNIVERSITY

This document is focus on Docker containers which have been created by the Earth Observation and Ecosystem Dynamics (EOED) research group (https://www.aber.ac.uk/en/dges/research/earth-observation-laboratory/) at Aberystwyth University (https://www.aber.ac.uk) to support our research. These containers are focused on the tools and software we use for our data analysis and therefore do not include all possible relevant tools but feel free to use the Dockerfile's as a basis to create your own containers.

Contents

1 Containers **5**
- 1.1 Why Containers? . . . 5
- 1.2 Where can Docker not be used? . . . 5
- 1.3 Installing Docker . . . 6
  - 1.3.1 MacOS and Windows . . . 6
  - 1.3.2 Linux . . . 6
- 1.4 Installing Singularity . . . 6
  - 1.4.1 Ubuntu . . . 6
  - 1.4.2 HPC . . . 7
- 1.5 Useful Images/Containers . . . 7
  - 1.5.1 au-eoed . . . 7
  - 1.5.2 spdlib . . . 7
  - 1.5.3 au-eoed-micmac . . . 8
  - 1.5.4 au-eoed-dev . . . 8

2 Docker: Running Data Analysis **10**
- 2.1 Docker Tools . . . 10
- 2.2 Container Terminal . . . 11
- 2.3 ARCSI . . . 11
- 2.4 Using GDAL Tools . . . 12
- 2.5 Running Python (RSGISLib) . . . 12

3 Singularity: Running Data Analysis **13**
- 3.1 ARCSI . . . 13
- 3.2 Using GDAL Tools . . . 14
- 3.3 Running Python (RSGISLib) . . . 14

4 Batch Processing Landsat with Docker and ARCSI **15**
- 4.1 Download Landsat Database . . . 15
- 4.2 Search Landsat for Scenes . . . 15
- 4.3 Setup Google SDK . . . 16
- 4.4 Download Landsat Images . . . 16
- 4.5 Generate ARCSI Commands . . . 16

4.6 Run ARCSI Commands with Docker 16

5 Conclusion **18**

Chapter 1

Containers

1.1 Why Containers?

The two main software for container are Docker (`https://www.docker.com`) and Singularity (`https://sylabs.io`). Docker it more widely used and is available for Linux, MacOS and Windows, however it needs to run as a root user. While singularity can be executed without root privileges but is only available for Linux systems (although a beta version is available for MacOS).

Docker (`https://www.docker.com`) is a lightweight container system which creates a stand-alone executable package including everything needed to run the package: code, runtime, system tools, system libraries, settings.

A single container can can then be executed on multiple platforms without making any changes to the software or having to re-build the software. The container can also be tagged to create a unique version of the system encapsulated within the container which can be reused at a later point ensuring that a result can be reproduced.

In this case, whether the software is a built via spack (`https://spack.readthedocs.io`) or conda-forge (`https://conda-forge.org`) we can encapsulate the tools and dependencies within a docker container for re-use.

1.2 Where can Docker not be used?

Docker cannot be used on a shared system (e.g., HPC) as it requires root access to execute which is not always possible or desirable. However, an alternative is available, which can import Docker containers, called Singularity (`https://sylabs.io/singularity/`) and this specifically supports HPC systems.

1.3 Installing Docker

1.3.1 MacOS and Windows

To install Docker, login into `https://hub.docker.com` and follow the instructions to download and install the software.

1.3.2 Linux

Ubuntu / Debian

Instructions are available from `https://docs.docker.com/install/linux/docker-ce/ubuntu/` but if you have not previously had Docker installed then the following commands are used:

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

CentOS

Instructions are available from `https://docs.docker.com/install/linux/docker-ce/centos/` but if you have not previously had Docker installed then the following commands are used:

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

Fedora

Instructions are available from `https://docs.docker.com/install/linux/docker-ce/fedora/` but if you have not previously had Docker installed then the following commands are used:

```
sudo dnf install docker-ce docker-ce-cli containerd.io
```

1.4 Installing Singularity

1.4.1 Ubuntu

```
sudo apt-get install singularity-container
```

1.4.2 HPC

If you are using a HPC system, such as Super Computing Wales (SCW) then singularity will need to be installed centrally, most likely via modules (`http://modules.sourceforge.net`) so you can loads singularity into your environment using a command such as:

```
module load singularity
```

1.5 Useful Images/Containers

NOTE Terminology: Docker has images and containers. A Docker Image is a set of files which have no state, whereas Docker Container is the instantiation of Docker Image. In other words, Docker Container is the run time instance of images.

On `https://hub.docker.com` you will find many useful Docker images which have been built and a ready to use.

The Docker images we will be using for the remainder of this tutorial are from myself and can be browsed at: `https://hub.docker.com/u/petebunting`.

1.5.1 au-eoed

This image contains the released version of RSGISLib (`https://www.rsgislib.org`) and ARCSI (`https://arcsi.remotesensing.info`) and their dependencies, such as GDAL (`https://gdal.org`).

This is the image which most will want to use for satellite imagery analysis.

URL

`https://hub.docker.com/r/petebunting/au-eoed`

Installation

```
docker pull petebunting/au-eoed
singularity pull docker://petebunting/au-eoed
```

1.5.2 spdlib

This image contains the released version of SPDLib (`https://spdlib`) its dependencies, such as GDAL (`https://gdal.org`).

This is the image which most will want to use for LiDAR data analysis.

URL

`https://hub.docker.com/r/petebunting/spdlib`

Installation

```
docker pull petebunting/spdlib
singularity pull docker://petebunting/spdlib
```

1.5.3 au-eoed-micmac

This image contains the released version of MicMac (`https://micmac.ensg.eu/index.php/Accueil`) its dependencies and scripts developed by the AU-EOED research group (`https://github.com/Ciaran1981/Sfm`) will allow processing of drone photogrammetry data.

This is the image which most will want to use for processing drone photogrammetry data.

URL

`https://hub.docker.com/r/petebunting/au-eoed-micmac`

Installation

```
docker pull petebunting/au-eoed-micmac
singularity pull docker://petebunting/au-eoed-micmac
```

1.5.4 au-eoed-dev

This image contains the development version of RSGISLib (`https://www.rsgislib.org`), ARCSI (`https://arcsi.remotesensing.info`) EODataDown and other software create by the AU-EOED group and their dependencies, such as GDAL (`https://gdal.org`).

This is the image should not be used in most cases as it changes regularly and at times may contain version of the software which are broken. However, it does contain the very latest versions of the various software and dependencies.

URL

`https://hub.docker.com/r/petebunting/au-eoed-dev`

Installation

```
docker pull petebunting/au-eoed-dev
singularity pull docker://petebunting/au-eoed-dev
```

Chapter 2

Docker: Running Data Analysis

2.1 Docker Tools

Once you have pulled your Docker image it is installed on your system, to see which images you have downloaded to your system using the following command:

```
docker image ls
```

To see which containers you have running you can use the following command:

```
docker ps
```

If you find that Docker is using a lot of storage space on your machine then the following command can be used to delete an image from your system:

```
docker rmi <IMAGE ID>
```

If you want to remove all components of the docker images/containers/volumes from your system then you can use the following command:

```
# Remove any images or containers which are 'dangling'
docker system prune
# Remove any images or containers which are 'dangling' or stopped.
docker system prune -a
```

2.2 Container Terminal

The simplest way to use the Docker image us to log into the container on a Terminal prompt. At this point your will have access to the software installed within the Docker image as you would from the Terminal on your own local machine. Running the following command will achieve this (NOTE: type `exit` to leave the container):

```
docker run -i -t petebunting/au-eoed /bin/bash
```

Once within the container try and run a command such as `gdalinfo --formats` to check the system is working.

However, you will noticed that you do not have access to your files, to get access to your local file system you need to mount this within the Docker container, as show below. NOTE: the variable `${PWD}` is a reference to the current location (i.e., where in your file system you have run the docker image from) this is being mapped on to the `/data` directory within the Docker container.

```
docker run -i -t -v ${PWD}:/data petebunting/au-eoed /bin/bash
```

From the terminal prompt within the Docker container you can now navigate to the `/data` directory, if you list the contents of the directory you will find the same files as where you execute the `docker run` command.

```
cd /data
ls -lh
```

You can also specify a specify local path to be mapped, for example:

```
docker run -i -t -v /scratch/MyCoolData:/data petebunting/au-eoed /bin/bash
```

Please note that you will now have to reference all your paths to `/data` and not the local paths on the machine you are working from. Also, all the data and scripts you want to use also need to be available in `/data`.

2.3 ARCSI

To run ARCSI using the docker image you use the same command as you would have otherwise done but you need to pre-append the Docker command and remember that all the files you are using are relative to the mount point within the Docker container.

```
docker run -i -t -v /scratch:/data petebunting/au-eoed arcsi.py -s ls5tm\
-p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT METADATA \
```

```
-o /data/Outputs/ --tmpath /data/tmp --dem /data/UKSRTM_90m.kea \
--k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
--stats --format KEA \
-i /data/Input/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

```
docker run -i -t -v /scratch:/data petebunting/au-eoed arcsi.py -s sen2 \
-p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT METADATA SHARP \
-o /data/Outputs  --dem /data/UKSRTM_90m.kea --tmpath /data/tmp \
--k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
--stats --format KEA \
-i /data/S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD.SAFE/MTD_MSIL1C.xml
```

2.4 Using GDAL Tools

Using one of the GDAL tools is similar to the ARCSI, in that the commands are all the same but you need to update file paths to be relative to the mount point in the Docker container. For example:

```
docker run -i -t -v /scratch:/data petebunting/au-eoed gdal_translate \
-of GTIFF /data/input_img.kea /data/output_img.tif
```

2.5 Running Python (RSGISLib)

Again, the change which is needed related to the file paths either being inputting into the python script. For example, the following python script (saved as `calc_ndvi.py`)

```
1 import rsgislib.imagecalc
2
3 img = '/data/landsat_img.kea'
4 out_img = '/data/landsat_ndvi.kea'
5 rsgislib.imagecalc.calcindices.calcNDVI(img, 3, 4, out_img)
```

Can be executed using the following Docker command:

```
docker run -i -t -v /scratch:/data petebunting/au-eoed \
python /data/calc_ndvi.py
```

Chapter 3

Singularity: Running Data Analysis

Singularity doesn't store images centrally but as files on your system.

```
-rwxr-xr-x 1 pete pete 753M  Sep 24 20:15 au-eoed-20190327.simg
-rwxr-xr-x 1 pete pete 1000M Sep 24 17:02 au-eoed-dev-20190921.simg
```

You can remove images by just deleting the files of the images you do no longer wish have on your system. I would recommend including the tag name (e.g., 20190921) in your singularity image name – you might need to rename the file once it has downloaded. You choose the image to run by using the full path to the image file.

3.1 ARCSI

To run ARCSI using the singularity image you use the same command as you would have otherwise done but you need to pre-append the Singularity command.

```
singularity exec /data/sw_imgs/au-eoed-20190327.simg arcsi.py -s ls5tm\
-p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT METADATA \
-o ./Outputs/ --tmpath ./tmp --dem ./UKSRTM_90m.kea \
--k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
--stats --format KEA \
-i ./Input/LT05_L1TP_203024_19950815_20180217_01_T1_MTL.txt
```

```
singularity exec /data/sw_imgs/au-eoed-20190327.simg arcsi.py -s sen2 \
-p CLOUDS DOSAOTSGL STDSREF SATURATE TOPOSHADOW FOOTPRINT METADATA SHARP \
-o ./Outputs  --dem ./UKSRTM_90m.kea --tmpath ./tmp \
--k  clouds.kea meta.json sat.kea toposhad.kea valid.kea stdsref.kea \
--stats --format KEA \
-i ./S2A_MSIL1C_20170617T113321_N0205_R080_T30UVD.SAFE/MTD_MSIL1C.xml
```

3.2 Using GDAL Tools

Using one of the GDAL tools is similar to the ARCSI, in that the commands are all the same, but you need pre-append the singularity command and image file. For example:

```
singularity exec /data/sw_imgs/au-eoed-20190327.simg gdal_translate \
-of GTIFF ./input_img.kea ./output_img.tif
```

3.3 Running Python (RSGISLib)

Again, the change which is needed related to the file paths either being inputting into the python script. For example, the following python script (saved as `calc_ndvi.py`)

```
1 import rsgislib.imagecalc
2
3 img = '/scratch/a.pfb/landsat_img.kea'
4 out_img = '/scratch/a.pfb/landsat_ndvi.kea'
5 rsgislib.imagecalc.calcindices.calcNDVI(img, 3, 4, out_img)
```

Can be executed using the following Singularity command:

```
singularity exec /data/sw_imgs/au-eoed-20190327.simg \
python /data/calc_ndvi.py
```

Chapter 4

Batch Processing Landsat with Docker and ARCSI

ARCSI has tools for bulk downloading and processing landsat and sentinel-2 data but require the Google Cloud SDK, the following commands can be used with two different docker images (`petebunting/au-eoed-dev` and `google/cloud-sdk`)

4.1 Download Landsat Database

The following command will download and build a local sqlite database for all the Landsat images collected globally:

```
docker run -i -t -v ${PWD}:/data petebunting/au-eoed-dev \
arcsisetuplandsatdb.py -f /data/lsgoog_db_20190924.sqlite
```

4.2 Search Landsat for Scenes

The following command searches the local database for path 227 and row 63 for collection 1 scenes and a cloud cover less than 50 percent.

```
docker run -i -t -v ${PWD}:/data petebunting/au-eoed-dev \
arcsigenlandsatdownlst.py -f /data/lsgoog_db_20190924.sqlite \
-p 227 -r 63 -o /data/ls_scns_dwnld.sh --outpath /data/ls_dwn \
--collection T1 --cloudcover 50 --multi --lstcmds
```

4.3 Setup Google SDK

If this is the first time you are using this google cloud sdk docker image, then you will need to authenticate it using the following command:

```
docker run -ti --name gcloud-config google/cloud-sdk gcloud auth login
```

4.4 Download Landsat Images

To download the scenes which were found through querying the database and outputted into file `/data/ls_scns_dwnld.sh` the following command:

```
docker run -ti -e CLOUDSDK_CONFIG=/config/mygcloud \
-v ${PWD}/mygcloud:/config/mygcloud -v ${PWD}:/certs \
-v ${PWD}:/data  google/cloud-sdk sh /data/ls_scns_dwnld.sh
```

4.5 Generate ARCSI Commands

The following command can be used to generate ARCSI commands for processing all the scenes which have been downloaded:

```
docker run -i -t -v ${PWD}:/data petebunting/au-eoed-dev
arcsibuildcmdslist.py -s LANDSAT -f KEA --stats \
-p CLOUDS DOSAOTSGL STDSREF --outpath /data/ls_ard \
--dem /data/srtm/srtm_3arc.kea --tmpath /data/tmp \
--keepfileends stdsref.kea clouds.kea -i /data/ls_dwn
-e "*MTL.txt" -o /data/ard_arcsi_cmds.sh
```

4.6 Run ARCSI Commands with Docker

The commands outputted do not have the docker command pre-appended so that needs to be pre-appended at the front of all the lines, so more cores can be used for the processing the following command can be used to split it into 4 output files:

```
docker run -i -t -v ${PWD}:/data petebunting/au-eoed-dev \
splitcmdslist.py -i /data/ard_arcsi_cmds.sh \
-p "docker run -i -t -v ${PWD}:/data petebunting/au-eoed-dev" \
-o /data/ard_arcsi_cmds_splt.sh -f 4
```

The output files can then be execute in different terminal windows using the following commands:

```
sh ard_arcsi_cmds_splt_1.sh
sh ard_arcsi_cmds_splt_2.sh
sh ard_arcsi_cmds_splt_3.sh
sh ard_arcsi_cmds_splt_4.sh
```

Chapter 5

Conclusion

Docker and Singularity are a really useful set of tools and solve a lot of challenges with software installation and deployment and transferring workflows between different machines (e.g., local laptop, workstation, HPC and Amazon or Google Cloud).

It is recommended that you develop your workflow and scripts on your local machine, probably using Docker, and then deploy onto the HPC or high performance workstation using Singularity.