**TIFFTiler Docker User Guide for PointBlue**

Tim DeBenedictis ([tdebenedictis-RA@pointblue.org](mailto:tdebenedictis-RA@pointblue.org))

Version 1.2 • 28 Nov 2022

These instructions describe how to install and run a Docker container for TIFFTiler. Later sections describe how to compile the native C++ tifftiler executable, and to create new Docker images from it.

**Caution:** as of this writing, these instructions have only been tested on Ubuntu 20.04, Windows 10 Pro (standard Point Blue issue), and MacOS. The docker image contains x86\_64 binary code, so will only run on a computer with a 64-bit Intel (or AMD?) processor.

**Installation**

First, install Docker on…

* **Mac:** <https://docs.docker.com/desktop/install/mac-install/>
* **Windows:** <https://docs.docker.com/desktop/install/windows-install/>
* **Linux:** <https://docs.docker.com/engine/install/ubuntu/>

Windows users: when installing Docker Desktop, select the WSL-2 option (not Hyper-V).

Test your docker installation by running

docker run hello-world

Next, pull the **tifftiler** docker image:

docker pull timmyd7777/tifftiler

**Running**

These instructions assume you have an orthomosaic on your local machine, and you want to save tiles to a different directory on your local machine. TIFFTiler does not (yet) pull an orthomoasic from an Amazon S3 bucket, or save tiles to another S3 bucket.

The following example assumes you are running on a Mac or Linux host. (There is a Windows example below):

* You have an orthomosaic (example) **croz\_20211127.tif** located in a directory **/home/ubuntu/Workspace/orthos**.
* Your tiles will be output to a directory **/home/ubuntu/Worspace/tiles**:

Here is the command to run the TIFFTiler docker container to generate tiles from that ortho (yes, this is all one line!):

docker run -it --mount src=/home/ubuntu/Workspace,target=/Workspace,type=bind timmyd7777/tifftiler /Workspace/orthos/croz\_20211127.tif /Workspace/tiles

Windows example (***note:*** use forward slashes, not backslashes, to separate directories!):

* You have the orthomosaic **croz\_20211127.tif** in a directory **c:/path/to/myDataDir/orthos**
* Your tiles will be written to **c:/path/to/myDataDir/tiles**

As with above, run the command (all in one line):

docker run -it --mount src=c:/path/to/myDataDir,target=/Workspace,type=bind timmyd7777/tifftiler /Workspace/orthos/croz\_20211127.tif /Workspace/tiles

Notes:

* The -it option means “interactive terminal”, which lets you type input and see output.
* The --mount describes the mapping between your host computer’s file system and the container’s filesystem. The **src** directory on your host computer (in the examples above, **/home/ubuntu/Workspace** or **c:/path/to/myDataDir**) is mapped to the **target** directory (**/Workspace)** inside the container.
* The name of the docker image is **timmyd7777/tifftiler**
* The next argument **/Workspace/orthos/croz\_20211127.tif** is the path to the orthomosaic *as seen by the docker container*, not the path to the orthomosaic on your host machine. (See above – this follows the logic of the mount)
* The last argument **/Workspace/tiles** is the path to the output directory for tiles, again as seen by the docker container. (See above – this follows the logic of the mount)
* If you don’t include the last two arguments, TIFFTiler will prompt you to enter them manually.

An example of successful output looks like this:

TIFF width: 182789

TIFF height: 171319

TIFF tile width: 0

TIFF tile height: 0

TIFF bits per sample: 8

TIFF samples per pixel: 4

TIFF planar config: 1

TIFF photometric: 2

TIFF compression: 5

TIFF scanline size: 731156

GDAL Geotransform:

169.207000, 5.119750e-07, 0.000000e+00

-77.446207, 0.000000e+00, -1.112440e-07

Opened input TIFF image file orthos/croz\_2020-11-29\_all\_col.tif.

Number of tiles x=372, y=726, total=270072

Read TIFF image strip 0!

Wrote 0 tiles; discarded 372 tiles.

Read TIFF image strip 1!

Wrote 0 tiles; discarded 372 tiles.

Read TIFF image strip 2!

Wrote 0 tiles; discarded 372 tiles.

Read TIFF image strip 3!

Wrote 0 tiles; discarded 372 tiles.

Read TIFF image strip 4!

Wrote 0 tiles; discarded 372 tiles.

Read TIFF image strip 5!

*etc etc etc.*

**Compiling TIFFTiler from C++ Sources**

These instructions are only relevant if you want to change the TIFFTiler program itself; don’t worry about anything below if you just want to run it.

The tifftiler executable is binary code compiled from sources written in C++. That C++ source is checked into the **counting\_penguins** repo, under **tiler/tiler\_cpp**. The source file TIFFTiler.cpp contains compilation instructions for Linux at the top. Here is a copy of those instructions.  
  
First install prerequisites:

sudo apt install build-essential libtiff-dev libgdal-dev

Then compile the executable (this is all one line!):

gcc -std=c++11 -o tifftiler TiffTiler.cpp -I/usr/include/gdal -L/usr/lib -ltiff -ljpeg -lgdal -lm -lstdc++

You can test the tifftiler executable by typing tifftiler. Command-line arguments are:

1. Path to input orthomosaic TIFF
2. Path to output directory to save tiles into

If you don’t enter any command-line arguments, tifftiler will prompt you for these items.

**Building the Docker Image**

First, install **dockerize**. This tool builds Docker images from binary executables (in ELF format, the standard for linux), including all of their dependent shared libraries. To install dockerize:

pip install git+https://github.com/larsks/dockerize

Then, to build a Docker image from the tifftiler executable (this is all one line!):

dockerize -t timmyd7777/tifftiler /home/ubuntu/Workspace/counting\_penguins/tiler/tiler\_cpp/tifftiler

Notes:

* **timmyd7777/tifftiler** is the name of the Docker image you want to create
* **timmyd7777** is my DockerHub account; please use your own preferred account
* The next argument is the full path to the tifftiler executable.

To push your Docker image up to DockerHub:

docker push

**Miscellaneous**

* The output tile sizes are hard-coded at 512x256 pixels.
* The output tile overlap is hard-coded at 20 pixels.
* The tifftiler executable is single-threaded; it wll not run any faster on a multi-core machine. Performance could be greatly improved by rewriting with a multi-threaded design.