# Docker Manual

This chapter will give you a rough overview of how to use the Docker containers installed on the UCC node.

- Getting Started with Docker on the REALTA Nodes

# Getting Started with Docker on the REALTA Nodes

# Getting Started with Docker on the REALTA Nodes

As of right now, all of the REALTA compute nodes (UCC*) and storage node (NUIG1) have some form of the standard I-LOFAR docker image. The NUIG1 machine has a slightly outdated version without GPU support (due to the lack of a CUDA device in that machine), but they should all contain the software you need to perform pulsar (or other realtime sampled data) processing.

The source of the image can be found here, and describes what software and which versions have been installed to the image.  While you are free to install or change software in the image at runtime (you have root access), if you have a request for an update to existing software, or something new you think should be in the global image, let David McKenna know.

## One Line Quickstart

If you just want to jump in and get started, here's a command you can use. It's advised to bind this to an alias to make it easier to call. It is current on the obs account under "dckrgpu".

```
# UCC Nodes
docker run --gpus all -e TERM -v /mnt:/mnt --rm -it pulsar-gpu-dsp2020


#NUIG Node
docker run -e TERM -v /mnt:/mnt --rm -it pulsar-dsp2020
```

These commands will launch a container container that will clean itself up when you exit a session.

As a result, it's advisible to run it within a tmux shell (`tmux new -S docker_workspace`) or screen (`screen -S docker_workspace`) to maintain it between work sessions. To deattached from these, you can use `Control+B,D` for tmux and `Control_A,D` for screen.

To exit a docker session, just use the standard bash `exit` command in the docker shell window.

# A Note on File Permissions

Any files created or modified within the docker container will become owned by the vritual root user. As a result, you may want to reclaim these files to your REALTA user account after you're finished in the container.

To do this, you will need to get your user ID and group ID on each node via the `id -u` and `id -g` commands (or `ll / ls -l` in a directory you own and look at the IDs). Afterward, you can reclaim the file via the `chown` command, running within any docker container:

```
root@b68e2344da1>:~/$ ls
my_working_file my_working_dir
root@b68e2344da1>:~/$ chown <realuid>:<realgid> ./my_working_file
root@b68e2344da1>:~/$ chown -R <realuid>:<realgid> ./my_working_dir/
```

# Other Flags

There are a few flags you might want to get familiar with and add in to the command above.

More `-v /ref/to/host:/mount/in/container` mounts can be of use, such as adding access to your home directory through the docker with `-v /home:/home_local`

`--cpuset-cpus="<procids>"` and `--cpuset-mems="<numa nodes>"` can be useful to ensure that your code is run in a single NUMA domain, given that `numactl` is not available within containers