Notes on running Jupyter Notebook & building Singularity images on the DCC cluster

- Create a Singularity container with a pre-built software environment that packages up all dependencies etc in an image (a Singularity container is similar to a Docker image but designed for data processing & suited to cluster use)
    - Provides web browser access to software such as Jupyter or RStudio

Overview & resources:

- Example Singularity container on the DCC:
    - https://gitlab.oit.duke.edu/OIT-DCC/singularity-example
    - Mike Newton made this example and is willing to answer questions
- Singularity container workshop through Duke Colab:
    - https://colab.duke.edu/roots/course/singularity-containers
- Useful intro notes on Singularity: https://www.vanderbilt.edu/accre/documentation/singularity/
- Singularity Desktop for MacOS: https://sylabs.io/guides/3.2/user-guide/installation.html?highlight=mac?highlight=mac#install-on-windows-or-mac
- Running jupyter notebooks with SLURM (setting up the SHH tunnel, etc):
    - https://alexanderlabwhoi.github.io/post/2019-03-08_jpn_slurm/
    - https://mjvo.github.io/tutorials/machine-learning/pytorch-dcgan/ (from a lab at duke / uses the DCC)

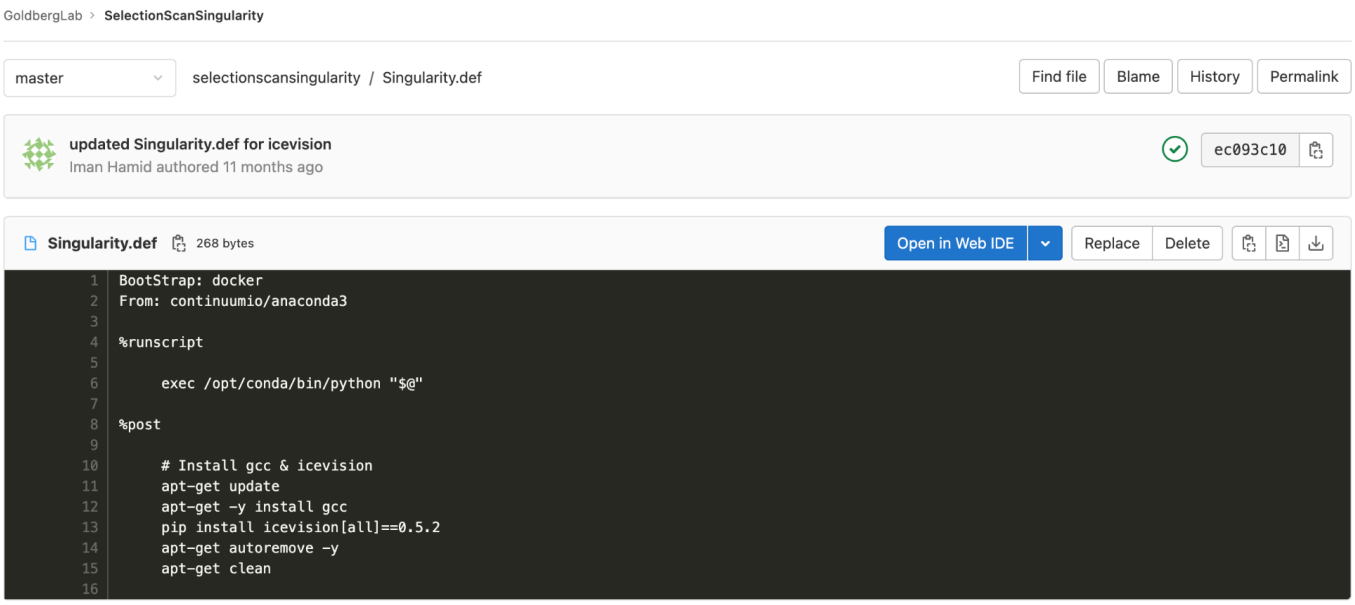
We created the group GoldbergLab in Duke's GitLab (GitLab is built for teams, more control over authentication levels): https://gitlab.oit.duke.edu/goldberglab

- Other members will need to be added by an admin using your netID. Login via Duke Shibboleth
    - For the example below, we made a project repository: SelectionScanSingularity
- Discussion with Mike
    - how do we allocate GPUs?
        - Add a "--nv" (can stack with the -B part of the command) to use nvidia GPU
    - How to use with jupyter notebook?
        - Ssh from login node into compute note
        - Open tunnel
        - Bind mount
        - Mike pointed out that this example will probably help https://gitlab.oit.duke.edu/OIT-DCC/rstudio-rstan

Example: building Singularity image for IceVision on DCC

Write a Singularity.def file to include installation of necessary software versions. Upload this to the project repository.

- Below, I wrote the Singularity.def file to include installation of IceVision v0.5.2. Also needed to install gcc in order for IceVision to install successfully.
    - We used example definition file for fastai2 found here: https://github.com/yijinlee/fastai2-def



Each time we push to the SelectionScanSingularity repository, it will automatically build the container image and upload it to https://research-singularity-registry.oit.duke.edu/<user or group id>/<repository_name>.sif

This build will take a few minutes. To check the status of the build you can go to https://gitlab.oit.duke.edu/<user or group id>/<repository_name>-/pipelines

It will tell you if it failed (and report errors) or if it was successful. Once the build is complete, you can pull down the image (replacing selectionscansingularity with your repository name):

```
curl -k -O https://research-singularity-registry.oit.duke.edu/goldberglab/selectionscansingularity.sif
```

(This build is already complete, so you can just pull the image assuming the environment is good to go!)

It's a couple gigs in size, so I wouldn't pull in your home directory.

chmod +x the .sif file to make it executable.

You can now run scripts with your predefined environment! E.g. from a worker node:

```
singularity exec -B /work yourcontainer.sif /work/netID/your_script.py
```

-B to bind mount a directory will be necessary if you are running any function or accessing data outside of your $HOME directory.

Running Jupyter Notebook via Singularity

These steps don't actually require GPU allocation (for future reference if anyone wants to run Jupyter Notebook on the cluster). You can run this from any of the compute nodes.

1. Request GPU or CPU allocation. e.g.:
   `srun -p gpu-common --mem=16G --gres=gpu:1 --pty bash -i`

(13.58 GB used when I trained the object detection model on 10k images.)

2. Set the runtime environment for Jupyter:
   `export XDG_RUNTIME_DIR=""`

3. Set port number:
   `export myport=19956`

   (can choose any set of 4 or 5 numbers you want)

4. From your **local machine**, open new Terminal window and set up tunnel:
   `ssh -NL 19956:dcc-core-gpu-03:19956 netID@dcc-login.oit.duke.edu`

   (replace 19956 with your port number, dcc-core-gpu-03 with the worker node you're on, and netID with your netID.)

   It will look like it's doing nothing / hung! This is normal.

5. From the compute node on the cluster, run jupyter notebook using Singularity image. e.g.:

```
singularity exec --nv -B /work selectionscansingularity.sif jupyter-notebook --no-browser --port=$myport --ip='0.0.0.0'
```

   I had to bind mount the /work path (using -B) since that's where I keep all the files for analysis. Bind mount will be necessary if you are running any function or accessing data outside of your $HOME directory.

   Make sure to properly define paths for any scripts or files you run using the bind path!

   --nv sets it to use the NVIDIA GPU, but not necessary if you're running on an ordinary CPU node.

6. You should see something that looks along these lines:

```
To access the notebook, open this file in a browser:
    file:///hpc/home/ih49/.local/share/jupyter/runtime/nbserver-1495059-open.html
  Or copy and paste one of these URLs:
    http://dcc-core-gpu-03:19956/?token=4592698d0e02a675dcaa8605ab35834406bf7b2c1cbeebb9
   or http://127.0.0.1:19956/?token=4592698d0e02a675dcaa8605ab35834406bf7b2c1cbeebb9
```

   Copy one of the ones with the token and replace dcc-core-gpu-03 or 127.0.0.1 with localhost and paste in your browser on your local machine (e.g. http://localhost:19956/?token=4592698d0e02a675dcaa8605ab35834406bf7b2c1cbeebb9)

Your Jupyter Notebook instance should be up and running from whichever directory you launched it from!