Getting Started with CARC for CSCI 662

CSCI 662

September 4, 2025

The purpose of this guide is to provide help with using USC CARC resources for your homework in CS 662. Further guidance will be provided on additional compute resources for the project. Throughout this guide, I use `ttrojan` as an example USCID - replace it with yours when you copy these commands. Direct CARC questions to Katy, not Jon, where possible.

Logging on to CARC

- You must be on either USC Secure Wireless or the USC VPN to access CARC.
- You will log onto CARC using `ssh`. If you don't already have an `ssh` key pair, see Github's documentation for more info.
- From a command-line environment on your local machine, do:

```
ssh ttrojan@discovery.usc.edu
```

- enter your USC SSO password. You shouldn't need to 2FA for this.
- You'll start out in your home directory on the `/home1` file system. Storage is limited here, and you should only use your home directory for personal configuration files and similar.
- To do your work for this class, navigate to our storage allocation on the `/project2` file system and make a user directory for yourself:

```
cd /project2/jonmay_1455
mkdir ttrojan
cd ttrojan
```

- Do your work in your directory **only** – do not mess with other students' files.
- When you log in, you start on a login node. It has no GPU, limited CPU, and is shared by a bunch of users. It's ok to edit code, do light debugging, or install packages from the login node. For GPU or CPU-intensive work, request an interactive session or batch job on a work node.
- All CARC nodes share the same filesystems, so your paths will work across nodes.

Transferring Files to and from CARC

- To transfer files to and from CARC, you will use the `hpc-transfer1.usc.edu` transfer node. You will run file transfer commands from your local machine, not from the CARC machine.
- I usually use `scp` for this:

```
# use scp -r for recursively copying directories
# local machine to CARC filesystem
scp /local/path ttrojan@hpc-transfer1.usc.edu:/project2/jonmay_1455/ttrojan/target
# CARC filesystem to local machine
scp ttrojan@hpc-transfer1.usc.edu:/project2/jonmay_1455/ttrojan/myfile
/local/target/
```

- You will need 2FA for the transfer nodes.
- CARC also provides documentation for using git and other file transfer tools including `sftp`, `rsync`, and others.

Editing Code on CARC

- You can edit code locally and use `git` to sync between local and CARC.
- You can also edit code directly on CARC using a command line editor like `emacs`, `nano`, or `vim`. I use `emacs`, so I can help with that but not `vim`.
- VSCode remote editing doesn't seem to work on CARC - it constantly drops the connection due to USC's timeout settings.

Using Python on CARC

- Autograding will run in Python 3.10, so we recommend using that on CARC too. To use Python on CARC systems, do:

```
module load python/3.10
```

- To install and manage python packages, you will need to use virtual environments. I recommend using a fresh venv for each HW. I personally use `pip` and `venv`; you can also use `conda`, but I am not familiar with it and won't be able to help debug.
- To create a new venv (from your user directory in our allocation):

```
python -m venv hw1/hw1-env
```

- Then, activate the virtual environment:

```
source hw1/hw1-env/bin/activate
```

- Install packages and run your code in the `venv`.
- If you want to use one `venv` for the whole class, then you may need to manually modify your `requirements.txt` to remove unnecessary dependencies before submitting to homeworks.
- For more info on virtual environments, see the documentation.

GPU	GPU Mem.	Good for...
p100	16 GB	anything BERT and smaller. More than enough for HW1. easy to get and usually available.
v100	32 GB	anything ~10B params or smaller, fairly easy to get and usually available
a40	40 or 80 GB	10B+ param models - may have longer queue times
a100	80 GB	10B+ param models - may have longer queue times
l40s	48 GB	10B+ param models - may have longer queue times

Table 1: GPU types available on Discovery.

Using GPUs on CARC

- To use GPUs on CARC, you'll need to request access to the resources using either `salloc` or `sbatch`. See following sections for more on these commands.
- Use the slurm options `--gres:gpu:type:num` to request resources and `-p gpu` to submit to the GPU partition.
- To see what GPUs are available, use this `sinfo` command:

```
sinfo -o '%20N %20G %10t' -p gpu
```

- you will need to load `gcc` and `cuda` to use GPUs/CUDA. I am currently using these versions in my own work.

```
module load gcc/13.3.0
module load cuda/12.6.3
```

Interactive Sessions on CARC

- To log into a GPU node interactively:

```
salloc --partition=gpu --ntasks=1 --gpus-per-task=type:num
```

- GPU nodes don't have internet access, so you'll need to download any external models/datasets/packages from the login node.
- Default time is (I think) one hour. For a longer allocation use `--time=x:00:00`

Submitting Batch Jobs on CARC

- To submit a job asynchronously, use `sbatch`:

```
sbatch myhomework.job
```

- Example job script:

```
#!/bin/bash

#SBATCH --account=jonmay_1455
#SBATCH --time=8:00:00
#SBATCH --gres=gpu:p100:1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=10GB
#SBATCH --partition=gpu

# I find emails useful, but this is personal preference - can remove these lines
#SBATCH --mail-user=ttrojan@usc.edu
# ALL will email you when the job is queued, when it starts,
# and when it finishes/times out/fails
#SBATCH --mail-type=ALL
#SBATCH --job-name=my_homework
# this will overwrite previous output files from previous runs of this job
#SBATCH --output=%x.out
# to keep all output files, put the job number in the output filename
#SBATCH --output=%j-%x.out

source /project/jonmay_1455/ttrojan/path/to/venv
module load gcc/13.3.0
module load cuda/12.6.3

CUDA_VISIBLE_DEVICES=0 #0,1 if using 2 GPU
python train.py --arg --arg --arg
```

Tips and Tricks

- CARC provides interactive Jupyter Notebooks via JupyterLab. See the CARC OnDemand documentation for more info. I personally haven't tried this, so I don't know much about it.
- useful aliases from my local `.zshrc`

```
# quickly log into discovery
alias discovery="ssh ttrojan@discovery.usc.edu"
# quickly log into discovery with port forwarding
# if you want to use tensorboard or similar.
# localhost:6006 on CARC will forward to localhost:16006 on your local machine
alias discovery="ssh -L 16006:127.0.0.1:6006 ttrojan@discovery.usc.edu"
```

- useful aliases from my `.bashrc` on Discovery

```
# quickly activate environment
alias envup="source /project2/jonmay_1455/ttrojan/some_env/bin/activate"
# quickly cd to your user dir for this class
alias cs662="cd /project2/jonmay_1455/ttrojan"
# see your jobs queued or running
alias q="squeue -u ttrojan -o '%.18i %.9P %.16j %.4u %.2t %.10M %.6D %R'"
```

```
# see your jobs queued or running, automatically updates every 10 secs. CTRL+C to exit.
alias qi="squeue -u ttrojan -i 10 -o '%.18i %.9P %.16j %.4u %.2t %.10M %.6D %R'"
# see which GPUs are available right now
alias sinfogpu="sinfo -o '%20N %20G %10t' -p gpu"
```

- `git-lfs` doesn't work well on CARC. If you need files larger than 100MB, just transfer them directly from your local machine.
- you can add `module load` statements to your `.bashrc` in your home directory to load them every time you log in

Resources and Help

- Katy's OH/slack DMs - please come to me before Jon with CARC questions
- SLURM Workload Manager documentation
- CARC User Guides and Documentation
- CARC User Forum – sometimes helpful, sometimes not so much
- CARC Office Hours