BlueHive Cluster Setup Guide

DSCC/LING 251/451 - Spring 2026

This guide covers everything you need to get started on the University of Rochester's BlueHive computing cluster. We'll cover this in class, but you can return to this document when you set up your own account.

Before You Start: Account Setup

You will need:

1. A BlueHive account (being requisitioned - you'll receive an email)
2. Two-factor authentication (2FA) enrolled
   - Enroll at: https://tech.rochester.edu/services/two-factor-authentication/
   - You'll use this every time you log in

Note: If you don't have your account yet, that's expected! Follow along today, and you can do the setup yourself once your account is active.

Part 1: Logging In

SSH Connection

From your terminal (Mac/Linux) or PowerShell (Windows), connect with:

```
ssh username@bluehive.circ.rochester.edu
```

Replace `username` with your University NetID.

What happens:

- You'll be asked for your password (it won't show as you type)
- Then you'll get a 2FA prompt (push notification or code)
- After authenticating, you're in!

Pro tip: On Mac/Linux, you can set up SSH keys to avoid typing your password every time. Ask if you're interested in setting this up.

Where Are You?

After logging in, you're on a **login node**. This is NOT where you run your code - it's just for:

- Navigating files
- Editing scripts
- Submitting jobs to the scheduler

Think of it like a lobby. You prepare here, but the work happens elsewhere (on compute nodes).

Part 2: File System and Storage

Two Important Directories

Run `pwd` to see where you are. You start in your home directory: `/home/username`

There are two main places to store files:

Directory	Path	Quota	Use for
Home	`/home/username`	10 GB	Config files (`.bashrc`, etc.)
Scratch	`/scratch/username`	200 GB	Data, code, conda environments, results

Why two directories?

- Home is backed up but tiny (10 GB)
- Scratch is large (200 GB) but NOT backed up
- Put everything except config files in scratch

Check your quota:

```
quota
```

This shows how much space you're using in each location.

Set Up Your Workspace

Create a directory structure in scratch for this course:

```
cd /scratch/username            # Go to scratch
mkdir -p dscc251                # Create course directory
cd dscc251                      # Enter it
mkdir data code environments    # Create subdirectories
ls -l                           # Confirm they exist
```

Navigation reminder:

- `cd path` - change directory
- `cd ..` - go up one level
- `cd ~` - go to home directory
- `cd -` - go back to previous directory
- `pwd` - print working directory
- `ls` - list files
- `ls -lh` - list with details and human-readable sizes

Part 3: Creating and Editing Files

You'll need to create scripts (Python, bash, etc.) on the cluster. Three main editors:

- **nano** - Easiest for beginners (we'll use this)
- **vim** - Powerful but steep learning curve
- **emacs** - Also powerful, different philosophy

Using nano

Create a test file:

```
nano hello.txt
```

Inside nano:

- Type your text normally
- `Ctrl+O` then `Enter` to save (Write Out)
- `Ctrl+X` to exit

Try it: Create a file with your name and save it.

Alternatives: If you prefer, you can write code on your laptop and use `scp` to transfer:

```
# From your laptop:
scp myfile.py
username@bluehive.circ.rochester.edu:/scratch/username/dscc251/code/
```

Part 4: Slurm - The Job Scheduler

What is Slurm?

BlueHive uses **Slurm** (Simple Linux Utility for Resource Management) to manage jobs.

Key concepts:

- **Login nodes**: Where you are now. For light tasks only (editing, navigating)
- **Compute nodes**: Where your code actually runs (GPUs, lots of memory)
- **Jobs**: Scripts you submit to Slurm that run on compute nodes
- **Partitions**: Groups of nodes (we'll use `standard` for this class; `gpu` partition is available if you need GPUs)

Why not just run Python directly?

- Login nodes are shared by everyone - running heavy code there slows things down for all users
- Compute nodes have GPUs and more resources

- Slurm ensures fair access to resources

Your First Slurm Job

Create a simple job script:

```
cd /scratch/username/dscc251/code
nano hello_world.sh
```

Contents of `hello_world.sh`:

```
#!/bin/bash
#SBATCH --job-name=hello          # Job name
#SBATCH --output=hello_%j.out     # Output file (%j = job ID)
#SBATCH --error=hello_%j.err      # Error file
#SBATCH --time=00:05:00           # Time limit (5 minutes)
#SBATCH --mem=1G                  # Memory
#SBATCH --partition=standard         # Partition

# Job starts here
echo "Hello from compute node!"
echo "Job ID: $SLURM_JOB_ID"
echo "Running on: $(hostname)"
echo "Current directory: $(pwd)"

# Simple Python test
python3 -c "print('Python works!')"
python3 -c "import sys; print(f'Python version: {sys.version}')"

echo "Job finished!"
```

Submit the job:

```
sbatch hello_world.sh
```

You'll see: `Submitted batch job 12345` (the job ID)

Check job status:

```
squeue -u username        # Your jobs
squeue -u username -l     # More details
```

Job states:

- `PD` (Pending) - waiting for resources

- R (Running) - currently running
- CG (Completing) - finishing up
- Nothing listed means it's done

View results:

```
ls -lh                     # See the output files
cat hello_12345.out        # Replace 12345 with your job ID
cat hello_12345.err        # Check for errors
```

Useful Slurm Commands

```
squeue -u username            # Your jobs
squeue -p standard               # All standard jobs
scancel 12345                 # Cancel job 12345
scancel -u username           # Cancel ALL your jobs
sinfo -p standard                # Partition info
sacct -u username             # Recent job history
```

Part 5: Setting Up Conda (Python Environments)

Why Conda?

- Manage different Python versions and packages per project
- Avoid conflicts between project requirements
- Create reproducible environments

Problem: By default, conda tries to install everything in your home directory (only 10 GB!). We'll configure it to use scratch instead.

Step 1: Load the Module

BlueHive uses "modules" to manage software. Load miniforge (conda):

```
module load miniforge3/25.1.1-2
```

Check it worked:

```
which conda    # Should show a path to conda
```

If you get "conda: command not found", you may need:

```
source /software/miniforge3/25.1.1-2/bin/activate
which conda    # Try again
```

Step 2: Make it Automatic (Add to .bashrc)

You don't want to run `module load` every time you log in. Add it to your `.bashrc` file:

```
nano ~/.bashrc
```

Add this line at the end:

```
# Load conda
module load miniforge3/25.1.1-2
```

What is .bashrc?

- A script that runs every time you start a new shell session
- Used for customizing your environment (aliases, loading modules, etc.)
- Lives in your home directory (`~/.bashrc`)

Activate changes:

```
source ~/.bashrc
```

Or log out and back in.

Step 3: Initialize Conda

First time only, run:

```
conda init --all
```

This modifies your `.bashrc` to set up conda.

Important: You need to run this BEFORE logging out. After running `conda init --all`, **log out and log back in** for the changes to take effect.

Verify it worked:

- You should see `(base)` at the start of your command prompt
- Run `which python` - should point to conda's Python

Step 4: Configure Conda to Use Scratch

By default, conda stores environments and packages in home. Move them to scratch:

```
# Create conda directories in scratch
mkdir -p $SCRATCH/my-conda/envs
mkdir -p $SCRATCH/my-conda/pkgs

# Tell conda to use these
conda config --add envs_dirs $SCRATCH/my-conda/envs
conda config --add pkgs_dirs $SCRATCH/my-conda/pkgs
```

Verify:

```
conda config --show envs_dirs
conda config --show pkgs_dirs
```

Should show your scratch paths listed first.

Step 5: Create an Environment

Create an environment for this course:

```
conda create -n dscc251 python=3.11 numpy pandas scikit-learn matplotlib
jupyter
```

Activate it:

```
conda activate dscc251
```

Your prompt should now show `(dscc251)` instead of `(base)`.

Check it:

```
which python        # Should be in your scratch conda env
python -c "import sklearn; print(sklearn.__version__)"
```

Deactivate when done:

```
conda deactivate
```

List all environments:

```
conda env list
```

Part 6: Using Conda in Slurm Jobs

When you submit a Slurm job, it starts a fresh shell that doesn't have your conda setup by default. You need to activate your environment in the job script.

Boilerplate for Conda + Slurm

Here's a template for Python jobs with conda:

```
#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --output=logs/job_%j.out
#SBATCH --error=logs/job_%j.err
#SBATCH --time=01:00:00
#SBATCH --mem=8G
#SBATCH --partition=standard
# #SBATCH --partition=gpu         # Use this partition if you need a GPU
# #SBATCH --gres=gpu:1            # Uncomment to request 1 GPU

# Print job info
echo "Job ID: $SLURM_JOB_ID"
echo "Node: $(hostname)"
echo "Start time: $(date)"

# Activate conda (module already loaded via .bashrc)
eval "$(conda shell.bash hook)"
echo "conda initialized"
conda activate dscc251
echo "environment activated"

# Verify environment
echo "Python: $(which python)"
echo "Conda env: $CONDA_DEFAULT_ENV"

# Run your code
cd /scratch/username/dscc251/code
python my_script.py

echo "End time: $(date)"
```

Key points:

- `eval "$(conda shell.bash hook)"` initializes conda in the job
- No need to `module load` since your `.bashrc` already does this
- `conda activate` works normally after the hook
- Echo statements help with debugging

- Always `cd` to your working directory before running code
- Create a `logs/` directory for output files

Test It

Create a simple Python script:

```
nano /scratch/username/dscc251/code/test_numpy.py
```

Contents:

```python
import numpy as np
import sys

print(f"Python version: {sys.version}")
print(f"NumPy version: {np.__version__}")

# Test computation
arr = np.random.rand(1000, 1000)
result = np.linalg.eigvals(arr)
print(f"Computed eigenvalues of 1000x1000 matrix")
print(f"First eigenvalue: {result[0]}")
```

Create a job script to run it:

```
nano /scratch/username/dscc251/code/run_test.sh
```

Use the boilerplate above, replacing `my_script.py` with `test_numpy.py`.

Submit:

```
mkdir -p logs  # Create logs directory
sbatch run_test.sh
```

Check the output file to see if it worked!

Quick Reference

Essential Commands

```
# Navigation
cd /scratch/username                  # Go to scratch
pwd                                   # Where am I?
```

```
ls -lh                              # List files with sizes

# Slurm
sbatch script.sh                    # Submit job
squeue -u username                  # Check my jobs
scancel 12345                       # Cancel job

# Conda
conda activate dscc251              # Activate environment
conda deactivate                    # Deactivate
conda list                          # Packages in current env
conda env list                      # All environments

# Modules
module load miniforge3/25.1.1-2     # Load conda
module list                         # What's loaded?
module avail                        # What's available?

# File editing
nano file.txt                       # Edit with nano
```

Where to Put Things

```
/home/username/           # Only .bashrc and small configs
/scratch/username/
    dscc251/
        code/             # Python scripts, job scripts
        data/             # Datasets
        logs/             # Slurm output files
        results/          # Model outputs, figures
    my-conda/
        envs/             # Conda environments
        pkgs/             # Conda packages
```

Troubleshooting

"conda: command not found"

- Did you run `module load miniforge3/25.1.1-2`?
- Did you log out and back in after `conda init`?
- Try: `source /software/miniforge3/25.1.1-2/bin/activate`

"Disk quota exceeded"

- Check: `quota`
- Is stuff in home instead of scratch? Move it: `mv ~/bigfile /scratch/username/`
- Clean conda cache: `conda clean --all`

Job pending forever

- Check: `squeue -u username -l` for reason
- `(Resources)` - cluster is busy, wait
- `(Priority)` - other jobs have higher priority
- Try a shorter time limit or less memory

Python can't find a package

- Are you in the right conda environment? Check the `(name)` in your prompt
- Did you install it? `conda list | grep packagename`
- In Slurm job: did you activate the environment? Check the boilerplate in Part 6.

Can't find my files

- Check `pwd` - are you in the right directory?
- Files in scratch? `ls /scratch/username/dscc251`
- Tab completion is your friend: type a few letters and hit Tab

---

Next Steps

Once your account is active:

1. Log in and verify 2FA works
2. Create your directory structure in scratch
3. Set up conda with the steps above
4. Run the hello world job to test Slurm
5. Create the `dscc251` environment
6. Test running Python with the numpy script

For the term project, you'll likely want to:

- Clone your GitHub repo to `/scratch/username/dscc251/code`
- Create a conda environment with your dependencies
- Write Slurm scripts to run experiments
- Transfer results back to your laptop for analysis

Questions? Post on the course forum or come to office hours!

Additional Resources

- BlueHive Documentation
- Slurm Documentation
- Conda Cheat Sheet
- Unix tutorial: https://swcarpentry.github.io/shell-novice/

Course-specific help: See the course website for examples of running specific ML libraries (PyTorch, scikit-learn, etc.) on the cluster.