Kamiak Cheat Sheet

Logging in to Kamiak

ssh *your.name*@kamiak.wsu.edu
ssh -X *your.name*@kamiak.wsu.edu — ***X11 graphics***

Transferring Files to and from Kamiak

From your laptop, not logged into Kamiak

scp -r *myFile your.name*@kamiak.wsu.edu:~ — ***Copy to Kamiak***
scp -r *your.name*@kamiak.wsu.edu:~/*myFile* . — ***Copy from Kamiak***
rsync -ravxS *myFile/ your.name*@kamiak.wsu.edu:~/myFile — ***Synchronize --sparse***

Linux Commands

cd	***Go to home directory***
cd ..	***Go up one level (.. is parent, . is current)***
cd ~/*myPath*	***Go to path relative to home (~ is home)***
ls	***List members of current directory***
pwd	***Show path of current directory***
mkdir -pv *myFolder*	***Create a directory (folder is synonymous with directory)***
cp -r -i *myFrom myTo*	***Copy file, -r for entire folder, -i prompt before overwrite***
mv -i *myFrom myTo*	***Move file or folder (-i to prompt before overwrite)***
rm -i *myFile*	***Delete file (-i to prompt before overwrite)***
rm -r -I *myFolder*	***Delete folder (-I prompt if delete more than 3 files)***
rm -r -f *myFolder*	***Delete entire folder, without asking***
rmdir *myFolder*	***Delete folder only if empty***
more *myFile*	***Display text file, one page at a time***
cat *myFile*	***Display entire file***
cat *myFile**	***Matches all files beginning with myFile***
du -hd 1 .	***See disk space on folder***
df -h .	***See disk space on volume***
man cp	***Manual page for command***
Ctl-c	***Kill current command***
Ctl-z	***Suspend current command***
bg	***Run suspended command in background***
fg	***Run suspended command in foreground***
disown -h	***Disconnect from terminal***

hpc.wsu.edu/cheat-sheet
hpc.wsu.edu/training/slides

Startup and Environment Variables

~/.bash_profile	***Executed on login and bash scripts***
~/.bashrc	***Executed on bash scripts***
PATH	***Search path for programs***
LD_LIBRARY_PATH	***Search path for libraries***
printenv	***List environment variables***
export variable=value	***Set environment variable***

Text Editors

vi
nano gedit
emacs

Snapshot Backups

/home/.snapshots/*myWSU.netid*/daily.2021-07-07_0000
/data/.snapshots/lab/*myLab*/daily.2021-07-07_0000 ***Backups over last 3 days***

Using Scratch Storage

export myScratch = "$(mkworkspace)"	***Create scratch folder, lifetime 2 weeks***
lsworkspace	***List scratch folders and expiration dates***

Submitting Batch Jobs to Kamiak

sbatch *myJob*.sh	***Submit a batch job script (to test, sbatch --test-only)***
squeue -u *your.name*	***View my pending and running jobs in the job queue***
squeue -j *jobNumber*	
scancel *jobNumber*	***Cancel a job***
sacct -S 2/26/18 -u *your.name*	***View job history including active jobs***
scontrol show job *jobNumber*	***View job details***

Viewing Information about the Cluster

sinfo -a | more	***What partitions and nodes are available***
squeue -a | more	***View all running and queued jobs***
scontrol show node cn93	***View node details (memory, cpus, GPUs)***

Interactive Session on Compute Node

```
idev -N 1 --ntasks-per-node=2 --mem-per-cpu=8G -t 360   #SBATCH same options
module load python
python -i
        print "Hello World!"
        exit()
srun -l python helloWorld.py
exit
```

Using Available Software on Kamiak

module avail	***Available modules compatible with compiler***
module list	***See loaded modules***
module spider	***See all modules***
module whatis anaconda3	***See what a module does***
module help wrf	***See help for a module***
module load python3/3.5.0	***Load specific version (Recommended)***
module load python	***Load latest or default version***
module unload python3	***Unload a module***
module swap intel gcc	***Replace intel with the gcc compiler***
module purge	***Unload all modules***
which python	***See that python is in your path***
printenv PATH	***See effects of loading modules on environment***
printenv LD_LIBRARY_PATH	

Sample Job Script

For multithreaded program that runs on 1 node

```
#!/bin/bash
#SBATCH --partition=kamiak          # Partition/Queue to use
#SBATCH --job-name=myJob            # Job name
#SBATCH --output=%x_%j.out          # Output file (stdout)
#SBATCH --error=%x_%j.err           # Error file (stderr)
#SBATCH --mail-type=ALL             # Email notification: BEGIN,END,FAIL,ALL
#SBATCH --mail-user=your.name@wsu.edu  # Email address for notifications
#SBATCH --time=7-00:00:00           # Wall clock time limit Days-HH:MM:SS
```

```
#SBATCH --nodes=1                    # Number of nodes (min-max)
#SBATCH --ntasks-per-node=1          # Number of tasks per node (processes)
#SBATCH --cpus-per-task=10           # Number of cores per task (threads)

module load python                   # Load software from Kamiak repository
srun python helloWorld.py -w         # Each task runs this program (total 1 times)
                                     # Each srun is a job step, and spawns ntasks
echo "Completed job $SLURM_JOBID on nodes $SLURM_JOB_NODELIST “
```

Sample Job Array

Template that spawns jobs, one for each array index

```
#!/bin/bash
#SBATCH --partition=kamiak           # Partition/Queue to use
#SBATCH --job-name=myJobArray        # Job name
#SBATCH --output=%x_%A_%a.out        # Output filename
#SBATCH --error=%x_%A_%a.err         # Error filename, group_index
#SBATCH --time=7-00:00:00            # Wall clock time limit Days-HH:MM:SS
#SBATCH --mail-type=ALL              # Email notification: BEGIN,END,FAIL,ALL
#SBATCH --mail-user=your.name@wsu.edu  # Email address for notifications

#SBATCH --array=0-2:1                # Number of jobs, in steps of 1
#SBATCH --nodes=1                    # Number of nodes (min-max)
#SBATCH --ntasks-per-node=1          # Number of tasks per node (processes)
#SBATCH --cpus-per-task=1            # Number of cores per task (threads)
#SBATCH --mem-per-cpu=8G             # Memory per core (gigabytes)

# Placeholder to create instances of a job as resources become available
# Creates 3 job instances, one for each index 1,2,3 ($SLURM_ARRAY_TASK_ID)
# Each instance is an individual job with the above resources
# Can use the index (in $SLURM_ARRAY_TASK_ID) in many ways
# Below the index splits data into 3 files: data_1.txt, data_2.txt, data_3.txt

module load python
srun python helloWorld.py "data/array_${SLURM_ARRAY_TASK_ID}.txt"
echo "Completed job array $SLURM_ARRAY_TASK_ID on host $HOSTNAME"
```

Types of Jobs

Single node ***Multithreaded program that runs on 1 node***

```
#SBATCH --nodes=1                          Threads share memory
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=20                 One core for each thread
export OMP_NUM_THREADS=$  SLURM_CPUS_ON_NODE  Optional if OpenMP
```

Multiple nodes ***Program runs once for each task on each node***

```
#SBATCH --nodes=2                          Tasks do not share memory, use MPI messages
#SBATCH --ntasks-per-node=2                Each task is a separate program instance
#SBATCH --cpus-per-task=10
```

GPU accelerator ***Program offloads kernel functions to GPU***

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4                One task per GPU (Graphics Processing Unit)
#SBATCH --cpus-per-task=1
#SBATCH --gres=gpu:tesla:4                 Number of GPU's per node
```

Getting Help

hpc.wsu.edu	***Support and Help-Desk Hours***
hpc.wsu.edu/training/slides	***Training slides***
hpc.wsu.edu/cheat-sheet	***This cheat sheet***

Appendix 1. Installing WSL (Ubuntu Linux) on Windows 10 or 11

Right-click on Command Prompt or PowerShell, to Run as Administrator.
Type: **wsl --install**
Restart
See: https://learn.microsoft.com/en-us/windows/wsl/install
WSL2 supports X11 GUI's natively without an X11 server.
You can also run ssh and scp under PowerShell without WSL.

Optional: Install common utilities

```
sudo apt update
sudo apt install zip
sudo apt install tcsh
sudo apt install dos2unix
sudo apt install bc
```

Optional: Set your home directory to your Documents folder

```
cp  ~/.profile  ~/.save.profile
WINDOWS_USERNAME=$(/mnt/c/Windows/System32/cmd.exe \
      /c 'echo %USERNAME%' | sed -e 's/\r//g')
cat <<-EOF >> ~/.profile
# Setup home directory
export HOME="/mnt/c/Users/$WINDOWS_USERNAME/Documents"
export DISPLAY=localhost:0.0
cd \$HOME
exec bash -l
EOF
```

Optional: Paste these commands so ssh does not forward locale

```
sudo cp  /etc/ssh/ssh_config  /etc/ssh/save_ssh_config
sudo sed -i 's/SendEnv/#SendEnv/' /etc/ssh/ssh_config
```

Optional: Install X11 Server (only needed on Windows if not running WSL2)

Install VcXsrv or XMing ***https://sourceforge.net/projects/vcxsrv/***
Launch Xlaunch *In dialog, use defaults*
ssh -X *your.name*@kamiak.wsu.edu ***You must start Xlaunch first***
For Mac: Install XQuartz ***https://www.xquartz.org/***

Tip: How to view pdf and images using X11

```
module load imagemagick
magick display myfile.png        # Or myfile.pdf
```

Appendix 2. Kamiak Bash Startup File (.bashrc)

Paste these commands to append common aliases to your .bashrc

```
cat <<-'EOF' >> ~/.bashrc
alias rm='rm -i'
alias mv='mv -i'
alias cp='cp -i --preserve=timestamps'
alias ls='ls -F -C'
alias more='less -i'
alias mkdir='mkdir -pv'
alias df='df -h'
alias du='du -h -d 1'
umask u=rwx,g=,o=       # only you can read and write (umask -S  to see settings)
EOF
```

Appendix 3. Installing your own Software

By default, software will try to install in the system libraries, to which you don't have write permission. Here's how to install software into your local environment. You do not need to install packages already installed on Kamiak.

Python local install

Do not use conda init (Not recommended)

It alters your .bashrc to always place you in conda

Create environment, and do local installs of packages into it

```
module load anaconda3  # Or miniconda3
conda create -n myenv
source activate myenv    # Only use "conda activate" if interactive
conda install whatever    # Ignore any "Failed to create lock" messages
```

Use environment in a script

```
module load anaconda3  # Or miniconda3
source activate myenv    # Only use "conda activate" if interactive
python                   # Watch out, python is python3 in anaconda3
```

Use Kamiak's installed conda apps in /opt/apps/conda/envs

```
module load miniconda3
module load conda_apps
source activate clair3-1.1.12
```

Install local packages using pip

module load anaconda3 # Or miniconda3
conda activate *myenv*
conda install pip
pip install *whatever*

Make python2 default in anaconda3

module load anaconda3
conda create -n python2 python=2.7
conda activate python2
python # This is now python2

Manage environments

conda config --show envs_dirs # Environments are grouped into paths
conda list -n root # See all available packages in miniconda3 path
conda env list # See list of my environments
conda list -n *myenv* # List packages in environment
source activate *myenv* # Activate environment
conda env config vars set somevar=value # Add bash export to env startup
conda deactivate # Deactivate environment
conda remove -n *myenv* --all # Delete environment

Sharing conda environments

conda create --prefix /data/lab/mylab/conda/envs/myenv
conda activate /data/lab/mylab/conda/envs/myenv
To add the mylab environment path to the conda search path:
(1) Edit: /data/lab/mylab/conda/startup/condarc
envs_dirs:
- /data/lab/mylab/conda/envs # appends to existing search path
env_prompt: ({name})
(2) Add to your .bashrc:
export CONDARC=/data/lab/mylab/conda/startup/condarc
(3) source activate myenv # No longer need absolute path

Shared installation using python virtual environments

```
module load python3
python3 -m venv /pathToPkg/env
/pathToPkg/env/bin/pip install \
        --install-option="--install-scripts=/pathToPkg/bin" cutadapt==2.7
export PATH=/pathToPkg/bin:$PATH     # To use cutadapt
```

Install packages into user's global environment (Not recommended)

```
module load python3
pip install --user whatever         # Install into ~/.local
pip3 install --prefix=~/myPython whatever   # Install into central location
export PYTHONPATH=~/myPythonlib/python3.5/site-packages:$PYTHONPATH
```

Perl local install

Type the following commands to append required setup to your .bashrc

```
echo  'module load perl'  >>  ~/.bashrc
echo  'eval $(perl -I$HOME/perl5/lib -Mlocal::lib)'  >>  ~/.bashrc
```

To install a package

```
cpan install someModule::somePackage       # Choose "manual" option for approach
```

R local install

Type the following commands to append required setup to your .bashrc

```
mkdir  -pv  ~/R/lib
echo  'export R_LIBS_USER=~/R/lib'  >>  ~/.bashrc
```

Using devtools in the Kamiak shared site library

```
export R_LIBS_SITE=/opt/apps/common/R/lib/%V/devtools
module load r                # R_LIBS_SITE is list of colon-separated library paths
R
.libPaths()
library("devtools")
```

To install a biocLite package

```
module load r  ;  R
biocLite("someApp", lib.loc="~/R/lib", lib="~/R/lib" )
```

Creating your own module files

Create a modulefiles folder

mkdir -pv ~/modulefiles/*myapp*
cp *3.1.lua* ~/modulefiles/*myapp*

Use your modulefile

module use ~/modulefiles
module load *myapp* # Searches your modulefiles in addition to Kamiak's

Example modules

module show gdal/2.3.1.lua # Examples in /opt/apps/modulefiles/Other

Manually add programs to your executable search path

cat <<-'EOF' >> ~/.bash_profile
PATH=$HOME*/apps/myapp*:$PATH
EOF

Appendix 4. Advanced Job Submission Techniques

Job Dependencies

$ sbatch job1.sh
11254323
$ sbatch --kill-on-invalid-dep=yes --dependency=afterok:11254323 job2.sh

Submitting to multiple partitions (do not mix kamiak and investor partitions)

#SBATCH --partition=cas,vcea # Lets the scheduler choose

Running on specific node or type of node

#SBATCH --nodelist=cn108
#SBATCH --constraint=avx-512 # Run on Xeon Scalable node

Pack jobs, chop up allocations and assign to different programs

#SBATCH -N 1 -n 2 --mem=384GB # pack-group 0, first component
#SBATCH packjob # (separator)
#SBATCH -N 1 -n 3 --mem=256GB # pack-group 1, second component
srun --pack-group=0,1 *myapp* # runs on both components (default is only on 0)

srun *myapp* : *myapp* # Alternative syntax to run on two components
srun --mpi=pmi2 : --mpi=pmi2 *myapp* # Can use with MPI
idev -N 1 -n 2 : -N 1 -n 3 # Can use interactively also

Appendix 5. Persisting Interactive Sessions

When using idev, to keep from disconnecting you can use tmux

ssh *your.name*@kamiak.wsu.edu
tmux new -s *myidev* # Run tmux on login node, not compute node
idev -N 1 -n 1 -t 360 # Run idev inside tmux, not the reverse
Ctl-b d # Detach

Reconnecting after getting disconnected

ssh *your.name*@kamiak.wsu.edu
tmux ls # Reconnect, must be on same login node
tmux attach -t *myidev* # Puts you back into the idev session on compute node
...commands
exit
exit

Make sure:

(1) You are on the same login node (login-p1n01 or login-p1n02).
If not, just ssh login-p1n01, or whichever login node you were on before.
(2) Run tmux on the login node, not on compute nodes.
(3) Run idev inside tmux, not the reverse.

Appendix 6. Troubleshooting

I can't transfer files from Kamiak onto my laptop, or from my laptop onto Kamiak

Remember to transfer files ***from*** your laptop, not in a window logged into Kamiak.
Just bring up a terminal window on your laptop, and then do:

```
scp -r ...
```

My program accidentally runs multiple times

Remember that srun runs its program once for each task (--ntasks times); for MPI this is once for each rank. For single-node multi-threaded programs, either omit the srun, or use --ntasks-per-node=1 and --cpus-per-task=20.

Seeing if your job is using cores

```
squeue -u your.name     # See where you are running
ssh cn14                # Log onto that compute node
htop                    # Core number is on left, program name is on right
                        # For memory bar, purple and yellow is for IO cache
                        # RES is memory in use, in kilobytes
                        # Hit F1 to see what the colors mean, q to quit
```

Seeing if your job is using GPU's

```
squeue -u your.name     # See where you are running
ssh sn3                 # Log onto that gpu compute node
nvidia-smi -l           # q to quit
```

My job fails with an out-of-memory error

Use --mem=240G or --mem-per-cpu=12G options of sbatch
to request more memory. (--mem=0 to request all the memory of a node).
To see how much memory you used (maxRSS is per task):

```
sacct -u your.name -o jobid,reqmem,maxrss,state,nodelist
```

My job gets cancelled due to preemption

Any job running in the backfill partition ("kamiak") can be preempted by an investor's job that needs the cores you are using on the nodes they own. Preempted means your job will be canceled and automatically resubmitted to the backfill queue to try again. You can reduce your chances of getting preempted by giving accurate resource requirements (e.g, fewer cores, shorter time).

XQuartz doesn't display graphical windows correctly on my Mac

Quit XQuartz, then in the macOS Terminal application (on your own machine, not logged into the remote cluster) run:

```
defaults write org.macosforge.xquartz.X11 enable_iglx -bool true
```

Restart XQuartz, start a new terminal window, and re-log into Kamiak using ssh -X or ssh -Y. This problem arises when using some OpenGL graphics features.

Appendix 7. Being a Good User

Don't

Do not run compute jobs or installs on a login node.
Use sbatch or idev to run them on a compute node.
Do not submit thousands of jobs – use job arrays.
Do not give your password to anyone, ever.

Do

Cite Kamiak in your work.
Report issues via Kamiak's Service Desk.
Abide by Kamiak's End User License Agreement (EULA) and WSU policies.
Use accurate resource requirements (CPU, time, memory).