Kamiak Cheat Sheet

Logging in to Kamiak

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

Transferring Files to and from Kamiak

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

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 *myFrom myTo*	***Copy file, -r for entire folder***
mv *myFrom myTo*	***Move file or folder***
rm *myFile*	***Delete file***
rm -r *myFolder*	***Delete folder***
rm -r -f *myFolder*	***Delete entire folder, without asking***
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***

Text Editors

vi
nano gedit
emacs

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
```

Using Scratch Storage

```
export myScratch = "$(mkworkspace -q)"   Create scratch folder, lifetime 2 weeks
export myScratch = "$(mkworkspace -q -b /local)"  Create scratch on /local SSD
```

Snapshot Backups

```
myFolder/.snapshot             Backups over last 3 days for /home and /data
```

Sample Job Script

```
#!/bin/bash
#SBATCH --partition=kamiak          # Partition/Queue to use
#SBATCH --job-name=myJob            # Job name
#SBATCH --output=myJob_%j.out       # Output file (stdout)
#SBATCH --error=myJob_%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=2         # Number of tasks per node (max)
#SBATCH --ntasks=2                  # Number of tasks (processes)
#SBATCH --cpus-per-task=1           # Number of cores per task (threads)

module load python                  # Load module from Kamiak repository
srun python helloWorld.py           # srun runs the program once for each task
                                    # (--ntasks times).   Each srun is a “job step”.
echo "Completed job $SLURM_JOBID on nodes $SLURM_JOB_NODELIST “
```

Sample Job Array

```
#!/bin/bash
#SBATCH --partition=kamiak          # Partition/Queue to use
#SBATCH --job-name=myJobArray       # Job name
#SBATCH --output=myJobArray_%A_%a.out       # Output filename
#SBATCH --error=myJobArray_%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 (max)
#SBATCH --cpus-per-task=1           # Number of cores per task (threads)
#SBATCH --mem-per-cpu=8G            # Memory per core (gigabytes)

# Runs this job 3 times, with index SLURM_ARRAY_TASK_ID as 0,1,2
# Split your data into 3 files, name them array_0.txt, array_1.txt, array_2.txt
# Each job array step is scheduled as an individual job
# Each job array step is allocated the above resources (cores, memory)

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"
```

Other Types of Jobs

MPI message passing

```
#SBATCH –nodes=2                    Multiple nodes, tasks do not share memory
#SBATCH –ntasks-per-node=2
#SBATCH –cpus-per-task=1
```

OpenMP shared memory

```
#SBATCH –nodes=1                    Single node, threads share memory
#SBATCH –ntasks-per-node=1
#SBATCH –cpus-per-task=20
export OMP_NUM_THREADS=$  SLURM_CPUS_ON_NODE
```

GPU (Graphics Processing Unit) accelerators

#SBATCH –nodes=1	***Offloads kernel functions to GPU***
#SBATCH –ntasks-per-node=4	***One task per GPU***
#SBATCH –cpus-per-task=1	
#SBATCH –gres=gpu:tesla:4	***Number of GPU's per node***

Getting Help

hpc.wsu.edu	***Support & Drop-in Hours***

Appendix 1. Installing Linux on Windows 10

Update Windows 10

Turn on Windows Subsystem for Linux

Launch Apps/Control Panel
In: Programs/Turn Windows Features on and off:
Turn on: Windows Subsystem for Linux

Install Ubuntu

Launch Apps/Microsoft Store
Search for Ubuntu, then install it
Launch Ubuntu
set your username and password
sudo apt update
sudo apt install zip
sudo apt install tcsh
sudo apt install dos2unix

Edit .profile to set home directory to your Documents folder

Launch Ubuntu
cd
cp .profile .save.profile # *.profile is in /home/YourName*
nano .profile # Add the following to the end, can use any editor such as vi

```
# Setup home directory
export HOME=/mnt/c/Users/yourName/Documents
export DISPLAY=localhost:0.0
cd $HOME
exec bash -l                # OR: exec tcsh -l
```

Install X11 Server

Install VcXsrv or XMing https://sourceforge.net/projects/vcxsrv/
Launch Xlaunch *In dialog, set display number to 0, others as default*
ssh -X *your.name*@kamiak.wsu.edu ***You must start Xlaunch manually***
For Mac: Install XQuartz https://www.xquartz.org/

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)

```
# Source global definitions
if [ -f /etc/bashrc ]; then
      . /etc/bashrc
fi
# User specific aliases and functions
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'
# Settings
umask 022   # only you can write, 002 group write, 007 group private, 077 private
```

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 in your local environment. You do not need to install packages already installed on Kamiak.

Python local install

Create environment, and do local installs of packages into it

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

Use environment

module load anaconda3
source activate *myenv*
python # Watch out, python is python3 in anaconda3

Install local packages using pip

module load anaconda3
source activate *myenv*
conda install pip
pip install *whatever*

Make python2 default in anaconda3

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

Manage environments

conda list -n root # See all available packages on Kamiak
conda env list # See list of my environments
conda list -n *myenv* # List packages in environment
conda remove -n *myenv* --all # Delete environment
source deactivate # Deactivate environment

Share environments across users

conda create --prefix */pathToMyenv/conda/envs/myenv*
source activate */pathToMyenv/conda/envs/myenv*

Install packages into user 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

Put the following in your .bashrc

module load perl
eval $(perl -I$HOME/perl5/lib -Mlocal::lib)

To install a package

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

R local install

Put the following in your .bashrc

export R_LIBS_USER=~/R/lib

To install a biocLite package

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

cd ; nano .bash_profile

PATH=$HOME/bin:$HOME/.local/bin:$HOME/apps/*myapp*/bin:$PATH

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

#SBATCH --partition=kamiak,cas # 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 (not recommended), or use --cpus-per-task=20 and --tasks-per-node=1.

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 packing your jobs into as few nodes as possible and giving a shorter request time.

Appendix 7. Being a Good User

Don't

Do not run intensive workloads on a login node (computing, installs, or long-running file transfers). Use sbatch or idev to run them on a compute node.

Do not submit thousands of jobs – use job arrays.

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).

Use /scratch and /local for your computational storage when possible.