UCDAVIS Bioinformatics Core

Working on UC Davis Bioinformatics Core Administrated Computing Systems

Matt Settles

Last Updated Aug Oct 3, 2017

Outline

- Introduction/Questions
- Explain user space (home directories) vs share space (lab directories/shares).
- Working in a team, file/folder permissions.
- The module system, when to request software, when to install software yourself.
- Slurm

UCDAVIS Bioinformatics Core

Quick Introduction to the UC Davis Bioinformatics Core

UCDAVIS Bioinformatics Core

The **mission** of the Bioinformatics Core facility is to facilitate outstanding omics- scale research through these activities:

Data Analysis

The Bioinformatics Core promotes experimental design, advanced computation and informatics analysis of omics scale datasets that drives research forward.

Research Computing

Maintain and make available high-performance computing hardware and software necessary for todays data-intensive bioinformatic analyses.

Training

The Core helps to educate the next generation of bioinformaticians through highly acclaimed training workshops, seminars and through direct participation in research activities.

UCDAVIS Bioinformatics Core

UC Davis Bioinformatics Core in the Genome Center

Core Facility Manager
Dr. Matthew Settles

Faculty Advisor
Dr. Ian Korf

Data Analysis Group

Genomics Bioinformatics

Dr. Joseph Fass
Dr. Monica Britton
Nikhil Joshi

Proteomics Bioinformatics

Metabolomics Bioinformatics

Dr. Jessie Li

Biostatistics

Dr. Blythe Durbin-Johnson

Undergraduate Assistant

Research Computing Group

System Administration

Michael Casper Lewis

Richard Feltstykket

Database/Web Programming

Adam Schaal

Undergraduate Assistant

UCDAVIS Bioinformatics Core

Contacts

- Computing Issues, include but not limited to

  User account questions, equipment failure/malfunction, software install, software failures (not related to use)

  helpdesk@genomecenter.ucdavis.edu

- Bioinformatics related questions, include but not limited to

  bioinformatic methods questions, software use, data questions

  Bioinformatics.core@ucdavis.edu

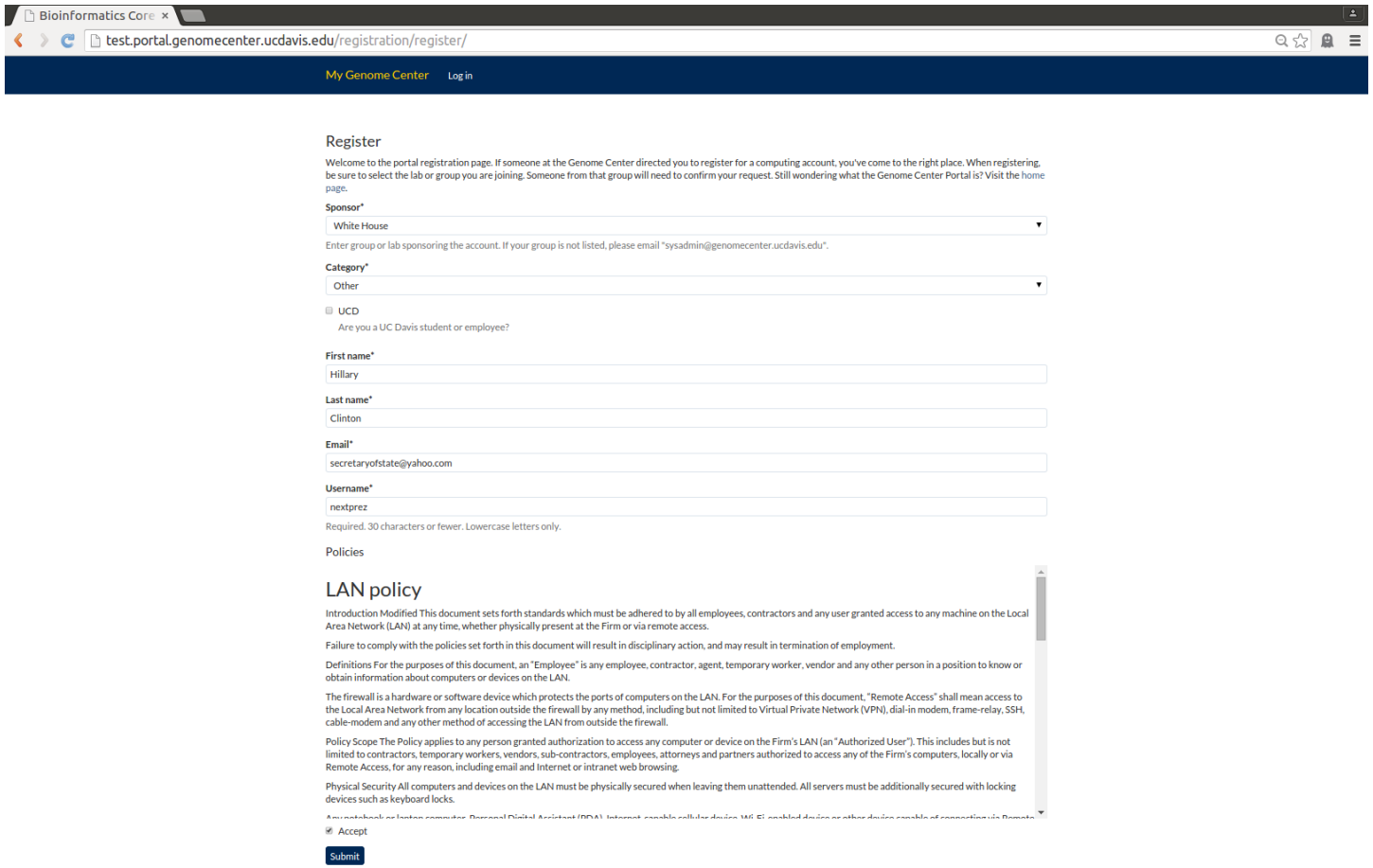
The ”Portal”

- UC Davis Bioinformatics Core Portal
  - Request users accounts
  - Manage groups/labs/access
  - Request software modules
  - Request Sys Admin Support
- Future
  - Integration with the new cluster, slurm/priority etc.
  - Storage accounts usage/backup, request more storage
  - View available modules and info (usage)

computing.genomecenter.ucdavis.edu

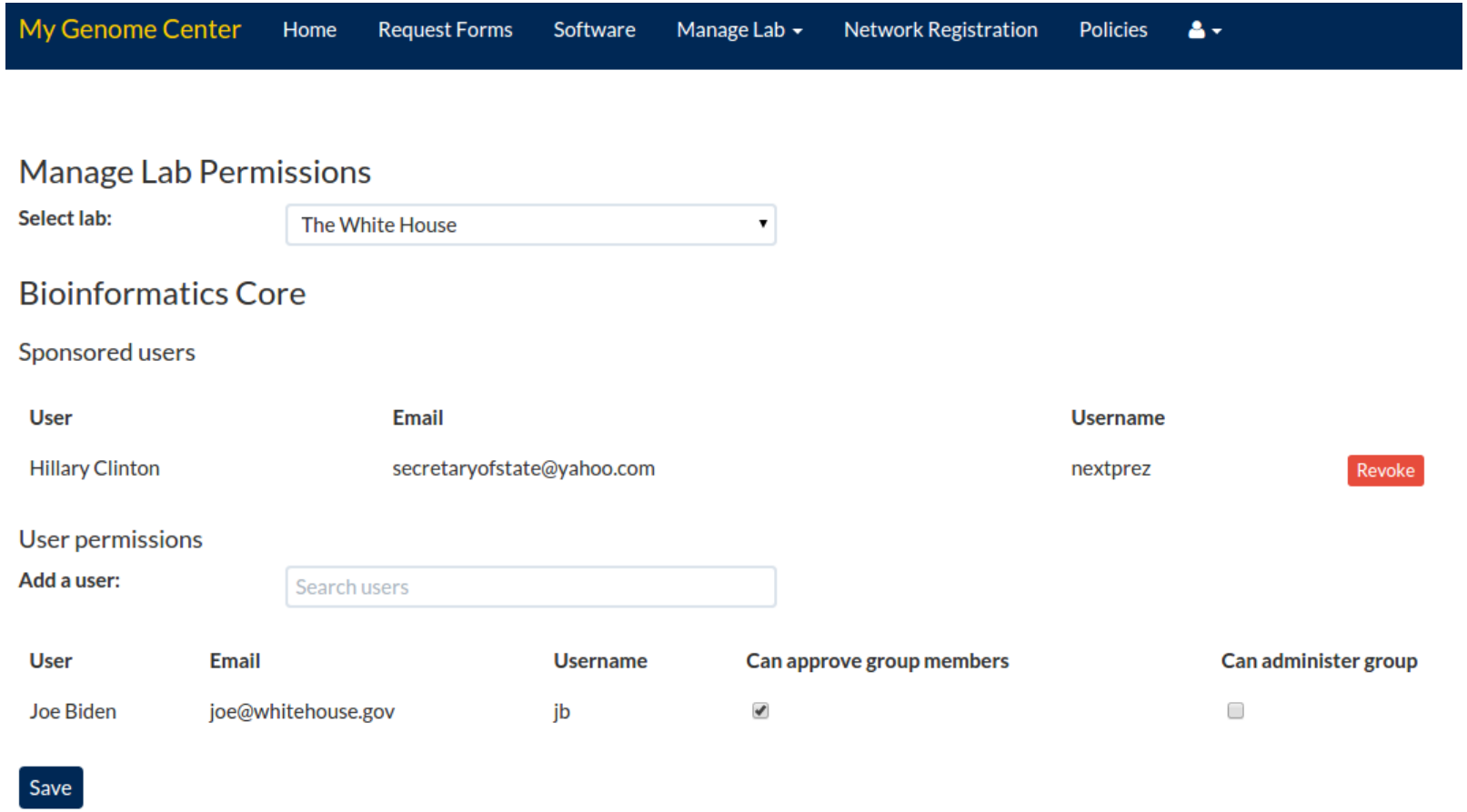
UCDAVIS Bioinformatics Core

Account registration



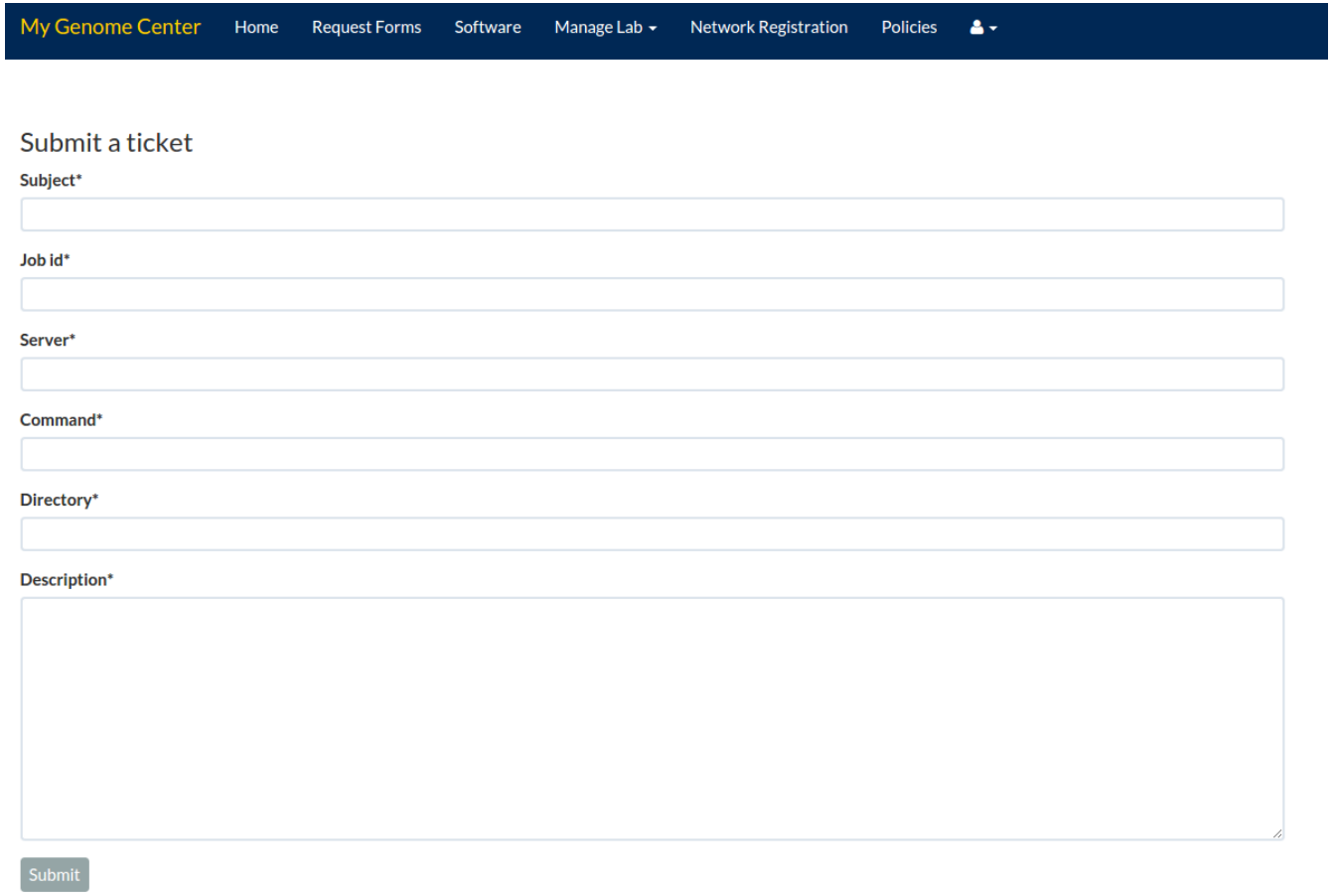
UCDAVIS Bioinformatics Core

Group memberships



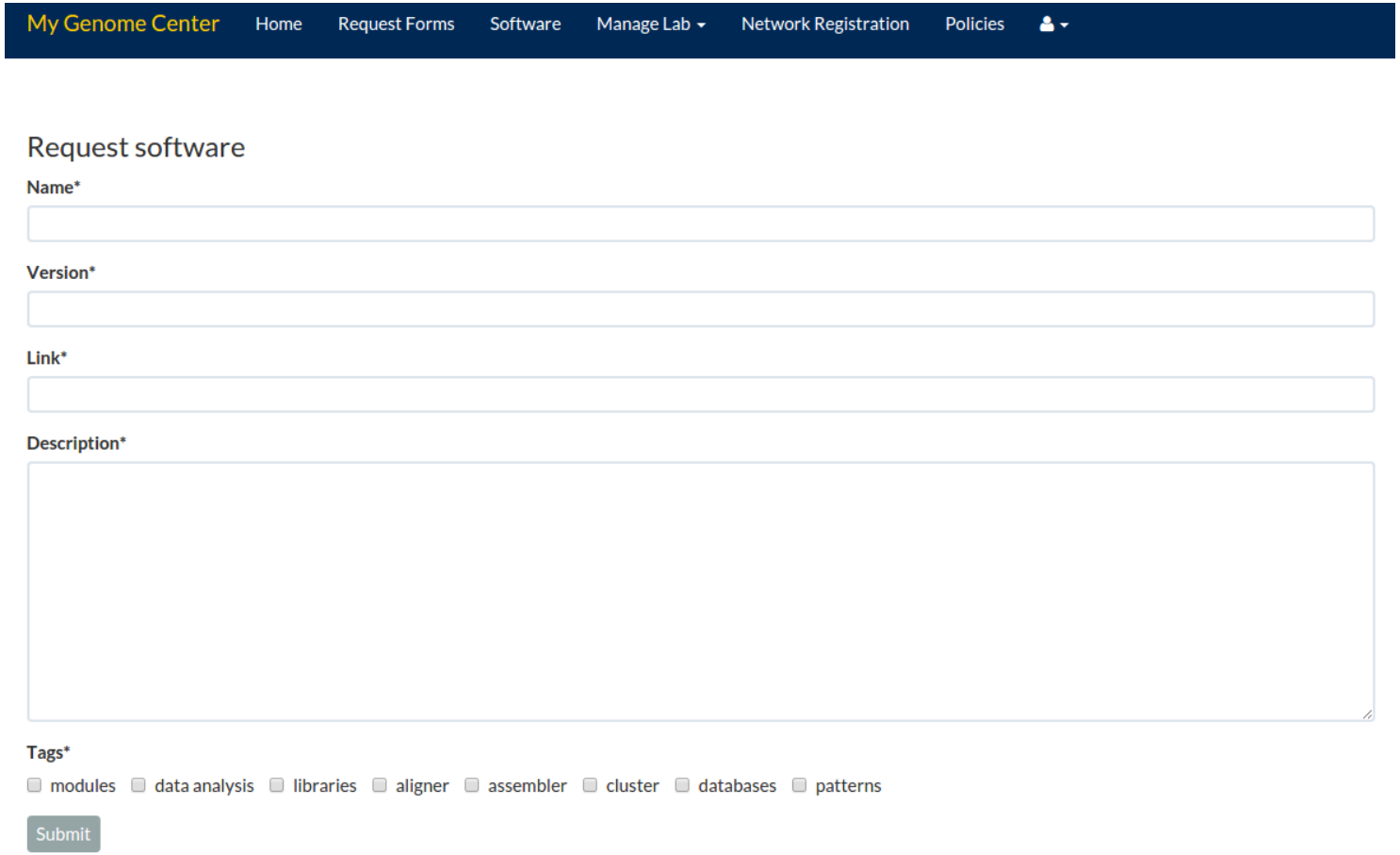
UCDAVIS Bioinformatics Core

Request help



UCDAVIS Bioinformatics Core

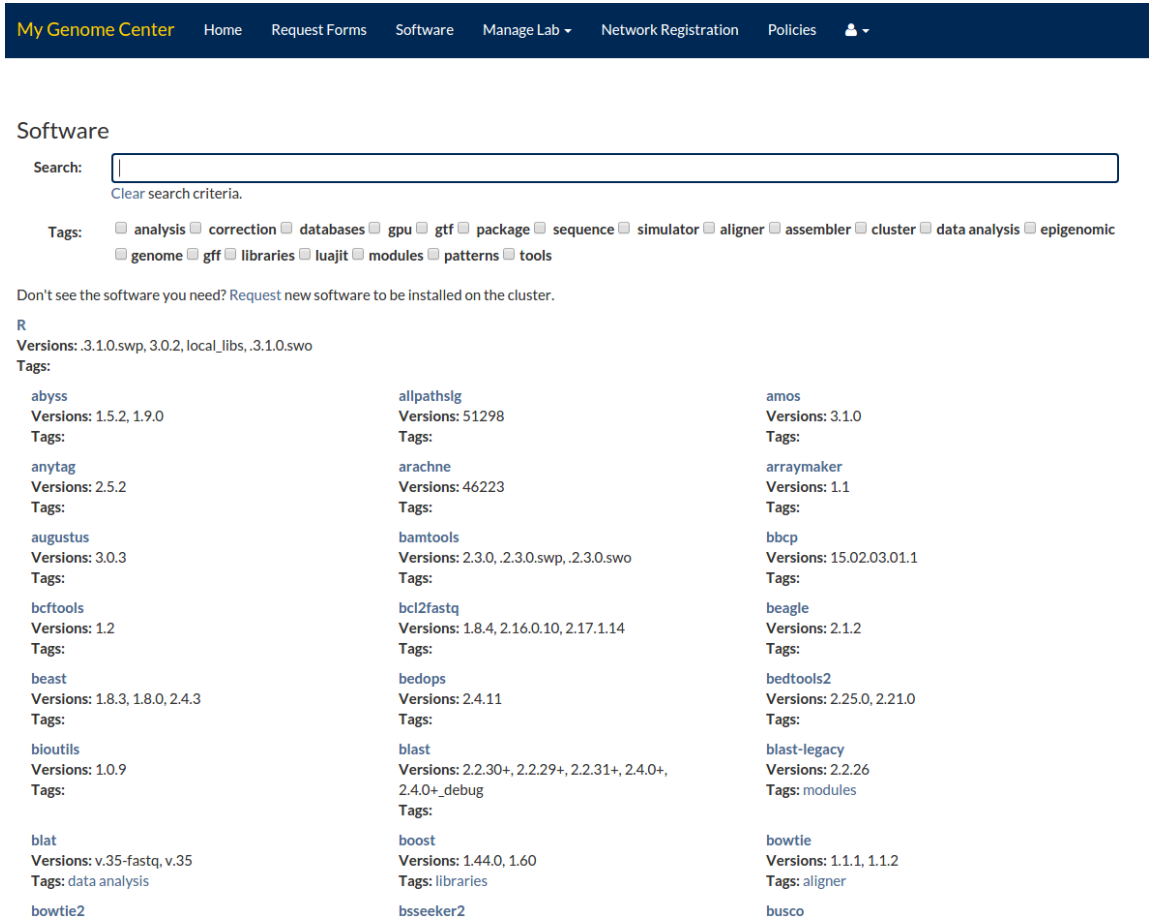
Software Request



UCDAVIS Bioinformatics Core

Software List

FUTURE



UCDAVIS Bioinformatics Core

Lab Share's

Home Directories

Cabernet

Slurm Scheduler

Lab/PI Server

HPC14

User space vs shared space
AND
Working in a team, file/folder permissions.
Groups/Labs, etc.

UCDAVIS Bioinformatics Core

What is a user account?

- User account are not tied to PI/labs. Any person who has a legitimate reason for requesting an account will be provided one. Users should only ever login as themselves and **never** as another user.
- The Bioinformatics Core provides users with a home directory that is capped at 100Gb (we are exploring reducing that to 10Gb).
- User home directories are ‘personal’ space, special permission must be given by university administration to explore a user’s home directory (while they are still at UC Davis).

Lab Groups

- The UC Davis Bioinformatics Core manages lab group data shares, top level folder associated with a collective group.

  These are data storage resources purchased by a lab, or PI.

- Groups/Labs/Collaborators work together in lab shares, found under /share/group_name

  [ex. The Bioinformatics Core share is /share/biocore]

- All lab associated analysis should be performed in lab shares. Virtually nothing should be done in your home directory, I personally have:
  - A link to my labs share
  - Software I've installed for me

Linux permissions

- Users on linux system have a user id and group(s) membership(s)
  - to see your group memberships on the command line type: groups
  - You may see an error, which can be safely ignored

    “groups: cannot find name for group ID 1107119953”
- Access to additional resources [data/compute] are driven by group permissions/memberships.
- More information on linux permissions

https://www.linux.com/learn/understanding-linux-file-permissions

By default

- By default, you have sole access to your home directory by both user and group.
- By default the lab share is owned by the PI and the group may be the PI group id, or a special group id [ex. subproject].
- By default, all shares are read accessible by everyone and write accessible by the user and the group.
- By default, the group id on a share is a ‘sticky’ bit, this means that any files/folders created under that share folder inherit the same permission (group read/write) but are owned by the user which created them.

Working as a team

- Groups and permissions are setup to facilitate team/group interactions.
- When working on files shared with other group members you don't have to request the other user to modify permissions, the group sticky bit takes care of that for you.
- YOU DO HAVE TO BE CAREFUL of others work! Since you have group write ability you are able to modify, delete and otherwise impact others people work/project.
- Once a file/folder has been created you can modify the permission to restrict access, however should the PI request the bioinformatics core to change it back, we will.

Centralized Resources

- Genomes – some genomes and mapping indexes (star/bwa/bowtie) are available centrally to be used by everyone, they can be found under
  - /share/genomes
    - Ensembl
    - genome_reference_consortium
    - Igenomes
    - Kraken
    - ncbi
    - ncbi_genomes
    - patric

UCDAVIS Bioinformatics Core

Module system

For Software

Modules

- The Environment Modules package provides for the dynamic modification of a user's environment via module files.
- Each module file contains the information needed to configure the shell for an application. Once the Modules package is initialized, the environment can be modified on a per-module basis using the module command which interprets module files. Typically module files instruct the module command to alter or set shell environment variables such as PATH, MANPATH, etc. module files may be shared by many users on a system and users may have their own collection to supplement or replace the shared module files.
- Modules are useful in managing different versions of applications. Modules can also be bundled into meta-modules that will load an entire suite of different applications, or ensure dependencies are loaded.

Modules

- Users can manage their environment (software available on the path) via modules.
- Users benefit from a familiar applications environment across UC Davis Bioinformatics Core systems (on and off-campus).
- Requests for software from any user becomes available to all users.
- Efficient system administration pooling software install/testing efforts from different projects/machines.
- Additional information
  https://en.wikipedia.org/wiki/Environment_Modules_(software)

UCDAVIS Bioinformatics Core

1. ssh

```
msettles@ganesh: ~$module avail

--------------------------- /software/modules/3.2.10/x86_64-linux-ubuntu14.04/Modules/versions ---------------------------
3.2.10

---------------------- /software/modules/3.2.10/x86_64-linux-ubuntu14.04/Modules/3.2.10/modulefiles ----------------------
abyss/1.5.2             bwa/0.6.2              gmap/2015-06-23        null                   samtools/1.3
abyss/1.9.0             bwa/0.7.10             gnumap/2.9.0           olb/1.9.4              samtools/default
allpathslg/51298        bwa/0.7.12             gnumaps/0.1            pantherscore/1.03      scythe/c128b19
amos/3.1.0              bwa/0.7.13             hisat/0.1.7-beta       pasa/r20140417         seqtk/1.0-r63-dirty
anytag/2.5.2            bwa/0.7.13-test        hmmer/2.3.2            pbccs/34288ba          sga/0.10.13
arachne/46223           canu/1.0               hmmer/3.1b1            pbcore/1.2.7           sickle/7667f147e6
arraymaker/1.1          cdhit/4.6.1            hmmer/3.1b2            pbdagcon/ed6e2c3       singularity/3a624f3
augustus/3.0.3          cdhit/4.6.4            homer/4.7              pbsuite/15.2.20        skewer/0.1.126-g88fb913
bamtools/2.3.0          ceas/1.0.2             htseq/0.5.3p9          pear/0.9.5             snap/2013-11-29
bbcp/15.02.03.01.1      cegma/2.5              htseq/0.6.1p2          penncnv/1.0.3          soapdenovo2/2.04-r240
bcftools/1.2            cgat/0.2.4             htslib/1.1             perl/5.14.4            spades/3.5.0
bcl2fastq/1.8.4         circos/0.67-1          jellyfish/1.1.11       perl/5.14.4-mt         spades/3.9.0
bcl2fastq/2.16.0.10     cmake/3.5.1            jellyfish/2.2.3        perl/modules-5.18.2    sprai/0.9.9.10
bcl2fastq/2.17.1.14     coral/1.4              kallisto/0.42.2.1      perl-modules/5.18.2    sratoolkit/2.4.2-3
bcl2fastq/default       crispresso/0.9.7       kentutils/308          plink2/1.90p           star/2.4.0h
beagle/2.1.2            cufflinks/2.2.1        khmer/1.1              poretools/0.5.1        star/2.4.2a
beast/1.8.0             cutadapt/1.8           khmer/2.0-rc1          pplacer/1.1.alpha18    star/2.5.1b
beast/1.8.3             dbg2olc/165ae2         kraken/0.10.5-beta     price/1.2              stringtie/1.0.4
bedops/2.4.11           delly/0.5.9            kraken/eaf8fb6         prodigal/2.6.3         subread/1.4.6-p3
bedtools2/2.21.0        delly/0.6.7            last/621               python/2.6.9           subread/1.5.0-p1
bedtools2/2.25.0        delly/0.7.2            leehom/dfca9e6         python/2.7.6-ubuntu    tabix/0.2.6
bioutils/1.0.9          deweylab-utils/0.0.2   lighter/1.0.6          qiime/1.9.1            tandem/15-04-01-1
blast/2.2.30+           diamond/0.7.9          macs2/2.1.0.20140616   quip/1.1.8             tophat/2.0.13
blast/2.2.31+           discovar/52218         mafft/7.245            R/local_libs           tophat/2.0.5
blast/2.4.0+            discovardenovo/52448   maker/2.31.8           randfold/2.0           tophat/2.1.0
blast/2.4.0+_debug      discovarexp/52219      masurca/2.3.2          ray/2.3.1              torrentsuite/4.6.11
blat/v.35               dot                    megahit/1.0.6          rdnatools/2a4cd4e      tpp/4.8.0
blat/v.35-fastq         dwgsim/0.1.11          meme/4.10.0_1          reapr/1.0.18           transdecoder/2.0.1
boost/1.44.0            ectools/0.1            meraculous/2.0.5       repeatmasker/4.0.5     trinity/2.0.6
boost/1.60              epimine/2016-01-13     meta/galaxy/1          rosetta/latest         trinity/2.1.1
bowtie/1.1.1            express/1.5.1          methprogs/19-Oct-2015  rsem/1.2.18            trinity/r20140717
bowtie/1.1.2            falcon/0.4.2           mira4/4.0.2            rsem/1.2.20            use.own
bowtie2/2.0.0-beta7     fastphase/1.4.8        mirdeep2/2.0.0.5       rsem/1.2.28            vcflib/7f1575e
```

Module basics

module avail, show, load, list, unload

$ module avail # shows the currently available module to the system # format is application/version

$ module show beast/1.8.3 # shows information, such as what-is, other modules loaded and path information

$ module load beast/1.8.3 # load the module beast version 1.8.3

$ module list # show currently loaded modules

$ module unload beast # unload the module beast

An Example

$ module list # no modules are loaded by default

$ module avail # lets see what modules are present, lets load beast

$ which beast # currently unavailable

$ module show beast/1.8.3 # show its load parameters

$ module load beast/1.8.3 # load beast v. 1.8.3 into our environment

$ module list # show the modules currently loaded

$ which beast # now we see beast is available on our path

We can now use beast

Modules are great for

- Modules are great for keeping track of what software and specifically what version of the software is being used in an analysis.
- You can keep track by
  - Doing an analysis from a bash script, loading the modules at the beginning of a script.
  - Copy/paste recording all commands to a file (ex. Analysis.txt) to save the commands for review later
  - Use history at the end of an analysis and save all commands including module load commands
- Note of caution: There are some applications available as a part of the system (ex. samtools) we are working to identify and remove these.

When to install software or request a module

- Benefits to having software in a module:
  - System administrators install, with troublesome software or difficult dependencies they can solve issues others can't.
  - Modules are available to all users, to so request benefits many
- Difficulties with software in a module:
  - Can take time for a system administrators to install, we manage over 400 servers, web servers, and multiple clusters, we do our best to be responsive.
  - If you're just trying something out that may or may not be useful can waste valuable system administrator time.

When to install software or request a module

- If you feel the software will be broadly applicable, request a module
- If it's a tough install, can't get it installed yourself, request a module
- If you just want to try something out, install it yourself
- If you need it right now, install it yourself

- If you installed it yourself AND requested a module, once the module is available delete your instance.

UCDAVIS Bioinformatics Core

Slurm scheduler

SLURM

Simple Linux Utility for Resource Management

- Manages compute resources
- Schedules jobs using those resources
- Original developed at Lawrence Livermore National Labs
- Open Source
- Supports plugins extending or enhancing functionality
- Increasingly being used at academic research computing centers and national labs
  - Berkeley
  - UCSD Super Computer

Slurm Commands

sinfo, squeue, sacct, srun, sbatch, scancel

sinfo reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.

squeue reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.

sacct is used to report job or job step accounting information about active or completed jobs.

srun is used to submit a job for execution or initiate job steps in real time. srun has a wide variety of options to specify resource requirements. A job can contain multiple job steps executing sequentially or in parallel on independent or shared resources within the job's node allocation.

sbatch is used to submit a job script for later execution. The script may contain one or more srun commands to launch parallel tasks.

scancel is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

Slurm Commands - SINFO

- **sinfo** reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.

  sinfo --summarize

- Node
- Nodes are computers/servers, multiple cpu with shared memory
- Partitions are logical groups of nodes (possibly overlapping)
  - gc64, gc128, gc256, gc512
  - Partition gc is the collection of gc64, gc128 and gc256 and is the default queue

Slurm Commands – SQUEUE

- **squeue** reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.
- Use squeue to view jobs currently queued and running
  - --user=username
  - --states=[running or pending]
  - --long # long format
  - --start # sometimes gives approximate start time

Slurm Commands -SBATCH

- **sbatch** is used to submit a job script for later execution [non-interactive]. The script may contain one or more srun commands to launch parallel tasks.

1. Login to cluster
2. Create a job script
   - I use a .slurm extension so that I know it's a job script
3. Submit the script to the cluster using sbatch
   - sbatch script.slurm
4. Slurm will respond with a jobid:
   - Submitted batch job 311533

- See example on github

Example Job script

```
#!/bin/bash
#SBATCH --partition=gc # partition to submit to
#SBATCH --job-name="myjob" # Job name
#SBATCH --nodes=1 # single node, anything more than 1 will not run
#SBATCH --ntasks=1 # equivalent to cpus, stick to around 20 max on gc64, or gc128 nodes
#SBATCH --mem=1G # memory pool all cores, default is 2GB per cpu
#SBATCH --time=0-01:30:00  # expected time of completion in hours, minutes, seconds, default 1-day
#SBATCH --output=my_output.out # STDOUT
#SBATCH --error=my_output.err # STDERR
#SBATCH --mail-user=you@gmail.com
#SBATCH --mail-type=ALL
# This will be run once for a single process
/bin/hostname
module load blat/v.35-fastq
echo "HELLO"
```

Slurm Commands – SCANCEL

- **scancel** is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.
- To delete a job
  - scancel <jobid>
  - Jobs not running yet (queued) are deleted, jobs running will be killed
- To delete many jobs
  - scancel <jobid1> <jobid2> <jobid3> …

Slurm Command – srun and helper scripts

srun is used to submit a job for execution or initiate job steps in real time. srun has a wide variety of options to specify resource requirements. A job can contain multiple job steps executing sequentially or in parallel on independent or shared resources within the job's node allocation.

An interactive session (preferred way)

The old script slurmlogin now serves as reminder on how to initiate an interactive session
msettles@cabernet: ~$slurmlogin
this script has been deprecated, for an interactive session run:
srun -t <timelimit> -n <number of cpus> --mem <ram> --pty /bin/bash

Examples:
srun --time 0:10:00 -n 1 --pty /bin/bash # 10 minute session with 1 cpu
srun --time 0:10:00 --exclusive --pty /bin/bash # a 10 minute session using the entire node,
srun –time 1-0 -p gc512 --mem 500000 –n 24 --pty /bin/bash
24 cpus, 500Gb of memory for a down on a 512Gb node

- If using ‘screen’,
  1. ‘screen’ on Cabernet,
  2. then srun,
  3. then exit when done.

Task Arrays

Job arrays offer a mechanism for submitting and managing collections of similar jobs quickly and easily; job arrays with millions of tasks can be submitted in milliseconds (subject to configured size limits).

- Slurm parameter, --array=a-b , where a and b are numeric number ex. 1-5
- Sets the environment variables
  - **SLURM_ARRAY_JOB_ID** will be set to the first job ID of the array.
  - **SLURM_ARRAY_TASK_ID** will be set to the job array index value.
- The value of $SLURM_ARRAY_TASK_ID can be used to reference rows in a file containing values to use for that job
- See example on github

Slurm command - sacct

- sacct - displays accounting data for all jobs and job steps in the Slurm job accounting log or Slurm database.
- Can be used like

sacct –j 275470 --format=jobid,elapsed,ncpus,ntasks,ReqMem,maxrss,MaxVMSize,state

```
Fields available:

AllocCPUS           AllocGRES           AllocNodes          AllocTRES
Account             AssocID             AveCPU              AveCPUFreq
AveDiskRead         AveDiskWrite        AvePages            AveRSS
AveVMSize           BlockID             Cluster             Comment
ConsumedEnergy      ConsumedEnergyRaw   CPUTime             CPUTimeRAW
DerivedExitCode     Elapsed             Eligible            End
ExitCode            GID                 Group               JobID
JobIDRaw            JobName             Layout              MaxDiskRead
MaxDiskReadNode     MaxDiskReadTask     MaxDiskWrite        MaxDiskWriteNode
MaxDiskWriteTask    MaxPages            MaxPagesNode        MaxPagesTask
MaxRSS              MaxRSSNode          MaxRSSTask          MaxVMSize
MaxVMSizeNode       MaxVMSizeTask       MinCPU              MinCPUNode
MinCPUTask          NCPUS               NNodes              NodeList
NTasks              Priority            Partition           QOS
QOSRAW              ReqCPUFreq          ReqCPUFreqMin       ReqCPUFreqMax
ReqCPUFreqGov       ReqCPUS             ReqGRES             ReqMem
ReqNodes            ReqTRES             Reservation         ReservationId
Reserved            ResvCPU             ResvCPURAW          Start
State               Submit              Suspended           SystemCPU
Timelimit           TotalCPU            UID                 User
UserCPU             WCKey               WCKeyID
```

Pipelines – using dependency

- sbatch returns the jobid after submission
- Can use that jobid and the sbatch argument --dependency to launch another job once the first one finishes

  --dependency=<type:job_id[:job_id][,type:job_id[:job_id]>

- See example on github

after:jobid[:jobid...]	job can begin after the specified jobs have started
afterany:jobid[:jobid...]	job can begin after the specified jobs have terminated
afternotok:jobid[:jobid...]	job can begin after the specified jobs have failed
afterok:jobid[:jobid...]	job can begin after the specified jobs have run to completion with an exit code of zero (see the user guide for caveats).
singleton	jobs can begin execution after all previously launched jobs *with the same name and user* have ended. This is useful to collate results of a swarm or to send a notification at the end of a swarm.

Coming soon

- Life Science Super Computer 0 (LSSC0)
  - Newer operating system
  - Newest version of slurm
  - Testing queue
  - Transfer node
  - Software install Nnode
  - New login node
  - Better accessibility to home directories, mounting of afs directories
  - Portal integration

UCDAVIS Bioinformatics Core

Contact

sysadmin@genomecenter.ucdavis.edu ; helpdesk@genomecenter.ucdavis.edu

software@genomecenter.ucdavis.edu

bioinformatics.core@ucdavis.edu

settles@ucdavis.edu

training.bioinformatics@ucdavis.edu