

Oscar

Oscar is Brown's high-performance computing cluster managed by the Brown [Center for Computation and Visualization](#) (CCV). Long-term visitors to ICERM with a Brown sponsored ID are provided with an exploratory account on Oscar upon arrival. Oscar access may also be requested by short-term visitors on a request-basis with advance notice.

- [Overview of Oscar](#)
- [SSH Access to Oscar](#)
- [Open OnDemand Access to Oscar](#)
- [Oscar Software Modules](#)
 - [Currently Available Modules](#)
 - [Oscar: Sage](#)
 - [Oscar: MATLAB](#)
 - [Oscar: Mathematica](#)
 - [Oscar: R](#)
 - [Oscar: Macaulay 2](#)
 - [Oscar: Magma](#)

Overview of Oscar

Long-term visitors to ICERM who receive a Brown ID are provided with an exploratory account on the Oscar high performance computing cluster maintained by Brown's [Center for Computation and Visualization](#). Oscar access may also be provided to short-term visitors on request-basis with *advance notice*.

If you are coming for a short-term visit and would like to request Oscar access, please contact ICERM IT staff no later than three weeks before your planned visit to ensure enough time to process a sponsored ID and enable Oscar access.

Oscar can be used via SSH terminal or via Oscar OnDemand through a web browser.

Using Oscar

Oscar provides both [command line \(SSH\)](#) access and VNC through [Open OnDemand](#) access to the computing cluster. All Oscar accounts are capable of using both interfaces, so you should choose the method that will work best for what you are trying to accomplish.

Oscar accounts are tied to Brown Shibboleth accounts, so you must have an ICERM Sponsored Brown ID to use Oscar. Once Oscar access is enabled for your Brown account, you can log in to CCV resources using your Brown Shibboleth username and password.

SSH Usage

Oscar's SSH interface provides a standard command line shell for interacting with the cluster. This is the best option for using simple command line scripts and submitting scripts for batch job processing. See the [SSH Login Instructions](#) for information on how to connect and basic usage. It's important to note that you should never run complex scripts or computations within the SSH login nodes. Computations should either be submitted as batch jobs or run in an interactive compute session using the `interact` command.

Open OnDemand

Oscar's OOD is a web portal to the Oscar computing cluster. This is the easiest way to access a number of CCV resources including an Oscar Shell, interactive applications like MATLAB, and a fully

featured Linux GUI. See the [OOD Login Instructions](#) for information on how to get started with the web portal.

Available Software

Oscar maintains a large library of software packages for use on the HPC cluster. Some of the most commonly used mathematical applications and languages available include:

- [MATLAB](#)
- [Mathematica](#)
- [Sage](#)
- Julia
- [R](#)
- Python
- Maple
- [Macaulay 2](#)
- [Magma](#)

To see the most up to date list of available software, log in to your Oscar account and run the terminal command `module avail`. When using Oscar over the Linux GUI, you must open the Terminal Emulator to run these commands.

Loading Software Modules

Oscar has a large library of software available on the cluster, but only a few apps are pre-loaded in to your sessions. The commands below will allow you to list all available modules, search the list of modules, and load/unload software packages.

- To view all available software packages, type `module avail`.
- To search the list of available packages, type `module avail <package>`.

For example, to search for all available versions of Mathematica command: `module avail mathematica`.

Many packages, like Mathematica, have multiple versions available. This command lets you see all available versions of the package you searched for.

- To load a package into your session, type `module load <package/version>`.
- For example, `module load mathematica/11.0`. This will load the Mathematica 11.0 into your session and make it available for use.
- To unload a package you are no longer using, type `module unload <packagename>`

- More information about Oscar's software packages are available in the [Oscar User Manual](#). If you require a software package that is not currently available on the Oscar cluster, please contact ICERM's IT staff and we will work with CCV to get the software installed.

Running Jobs

Oscar supports two main methods of running jobs: **batch jobs** and **interactive sessions**.

- Batch jobs are pre-scripted and can be submitted to the cluster's scheduler via the `sbatch` command.
- Interactive jobs are command-line sessions run directly on a compute node via the `interact` command that can be used in real time.

The [Running Jobs](#) page of the [Oscar User Manual](#) provides the most detailed and up-to-date instructions on scripting and submitting jobs.

Important Notes About the Oscar Cluster

- **Please do not run any computations or simulations on the login nodes, as they are shared with other users. Use SLURM to submit a batch job to the queue for computations or start an interactive session on one of the compute nodes with the command `interact`.**
- The full [Oscar User Manual](#) is available on the CCV website.
- Oscar uses SLURM for managing batch jobs and interactive sessions on the cluster. Detailed instructions on submitting jobs is available on the [Running Jobs](#) page of the [Oscar User Manual](#).
- Users can install sub-packages for some modules (like Sage and Python) to their home folders on their own by using the `-user` as an option in the install command. For example, with Sage, the command would be something like `sage -i <package name> -user`.

If you have questions about these instructions or require further assistance, please contact the ICERM IT staff by dropping by the administrative offices or emailing support@icerm.brown.edu.

SSH Access to Oscar

1. Open your preferred terminal application. Windows users need to install an SSH client. We recommend PuTTY, a free SSH client for Windows.
2. In the terminal, type `ssh <your ccv username>@ssh.ccv.brown.edu`. If you are asked to verify the authenticity of the host 'ssh.ccv.brown.edu', type `yes`.
3. You will now be prompted for your password. Enter your password (nothing will show up when you type in the terminal password prompt) and press enter.
4. Once logged in, you should see a "Welcome to Oscar!" message. This means you're now connected to one of the login nodes, which you can use to manage your files and submit batch jobs.

Additional Information:

1. To authenticate to Oscar with SSH keys, refer to this [documentation](#).
2. Please DO NOT run any computations directly on the login nodes. Use the batch system to submit your computations to the queue to be processed on the computation nodes or start an interactive session on one of the compute nodes with the command `interact`. See CCV's Oscar Documentation page on [Running Jobs](#) for detailed instructions on batch jobs and interactive sessions.

Open OnDemand Access to Oscar

Open OnDemand (OOD) is a web portal to the Oscar computing cluster. **An Oscar account is required to access Open OnDemand.** Visit this [link](#) in a web browser and sign in with your Brown username and password to access this portal.

Benefits of OOD:

- No installation needed. Just use your favorite browser!
- No need to enter your password again. [SSH into Oscar](#) in seconds!
- No need to use two-factor authentication multiple times. Just do it once, when you log into OOD.
- Workflow remains the same with or without VPN.










Once logged in you'll see a landing page that looks similar to the photo below:

OPEN

OnDemand

OnDemand provides an integrated, single access point for all of your HPC resources.

Pinned Apps A featured subset of [all available apps](#)

Default GUIs			
 Active Jobs System Installed App	 Home Directory System Installed App	 Job Composer System Installed App	 OSCAR Shell Access System Installed App
 Basic Jupyter Notebook with Anaconda System Installed App	 Desktop System Installed App	 Jupyter Notebook for Python Environments System Installed App	 MATLAB on OSCAR System Installed App
			

Guides to help you figure your way out once logged in:

[Using File Explorer on OOD](#)

[Web-based Terminal App](#)

[Interactive Apps on OOD](#)

[Using Python or Conda environments in the Jupyter App](#)

Oscar Software Modules

Below are instructions for software modules available on the Oscar clutter.

Currently Available Modules

If you require a software package that is not currently available on the Oscar cluster, please [contact ICERM's IT staff](#) and we will work with CCV to get the software installed.

This list is current as of May 6, 2025. Please scroll across the code snippet to see the full list. For the most up-to-date list of software modules, log into your Oscar account and run the command `module avail`.

Note: D is the default module.

abaqus-container/2021-akaeexs	filezilla/3.49.1-epfjuus	mafft/7.505-
iuicuiv	qit/2023-04-04-grwuvvgg	
abaqus/2017-q4ghhm5	flashpca/2.0-zr2wflq	magma-usyd/V2.28-8-
p3zylpg	qmcpack-mpi/3.16.0s-qp2hymx	
abaqus/2021.1-i675dvw	freebayes/1.3.6-v7rppcd	magma/2.7.1-
cpueyjy	qscintilla/2.11.6-dq7zlcq	
abaqus/2024-h5273a3	(D) freeglut/3.2.2-76qqoqn	maple/22-
cp2uld4	qt/5.15.9-fb7mjex	
abaqus/2024-ir-a4m5ld5	freesurfer/7.3.2-zop5n6m	mark/2018.07.08-
43snzfu	qualimap/2.2.1-mybpdoi	
abaqus/2024-mbessa-ceayfuo	freesurfer/8.0.0-jotmypd	(D) materialstudio/2024s-
gbc3dg6	quantum-espresso-mpi/7.1-gits-v4bxgtv	
abaqus/2024.1-7pcdqhp	fsl/6.0.7.7s-bul4mby	mathematica/13.2.0-
n4i6yua	quantum-espresso-mpi/7.1s-ia43pjk	
admixture/1.3.0-onwaqrp	fv/5.5.2-g2ibb5x	matlab/R2019a-
rjyk3ws	quantum-espresso-mpi/7.3s-kydgjwo (D)	
afni/23.3.07s-zm43m3u	fzf/0.45.0-pdwl7a4	matlab/R2023a-
xd6f7ph	(D) r/4.0.0-p7gxu4e	
afni/24.2.01s-kstpoqt	(D) gatk/4.3.0.0-234wqft	matlab/R2024b-
ipaztju	r/4.0.3-pvf2znb	
anaconda/2023.09-0-7nso27y	gaussian/09_v1-u6klkps	maven/3.8.4-
w3zgh4v	r/4.1.0-bfjsvw5	
angsd/0.935-cbhuwc7	gaussian/09-D01_v2-tw73726	mercurial/5.8-
vly6btb	r/4.2.2-z6qdiis	
ant/1.10.13-alpqj4j	gaussian/09-D01-TEST_v3-vv6ar67	mesa/22.1.6-
yi2tztm	r/4.3.1-lmofgb4	

ants/2.4.3-75npypop	gaussian/16-C01-bb2r2gh	(D) meson/1.6.0-
eipcwzq	r/4.4.0-yyccctsj	
aria2/1.36.0-lsb7zcs	gaussview/v05-mkdyw6j	metis/5.1.0-
7qoahod	r/4.4.2-re5rjx3	(D)
arm-forge/22.1.3-zq7lvdq	gcc/6.5.0-lwshmx	miniconda3/23.11.0s-
odstpk5	raisd/2.9-svyic22	
armadillo/12.2.0-4clpczv	gcc/10.1.0-mojgbnp	miniforge/23.11.0-0s-
hwmjdtj	rclone/1.62.2-o4lkrv6	
atom/1.19.3-ty5sdsn	gcc/13.1.0-nvrtbp3	(D) minimap2/2.14-
33hmvx2	readline/6.3-rnqups2	
autoconf/2.69-p4rpx2	gcm/2.4.1-lfqoahr	molden/6.7-
isryqwj	readline/8.2-5xbuyjt	(D)
autoconf/2.71-opdgqng	(D) gdal/3.7.0-4p4onmf	molden/7.3-
kytvh3m	(D) root/6.28.04-u7t5ax7	
avogadro2/1.99.0-5zl5qaw	gdal/3.10.3-4vc4f76	(D) molpro-mpi/2023.2.0s-
vyfv74n	rsem/1.3.3-5obucw6	
awscli/1.27.84-v22kngs	geeque/2.4-6vdnc4v	molpro-mpi/2024.3.1-mpipr-
hwakoux	(D) rstudio/2023.09.1-lsqy746	
bamtools/2.5.2-ki3mdef	geos/3.11.2-a6hfu6a	mpc/1.3.1-
zbino7j	ruby/3.1.0-gnoxsfm	
basilisk/2023.11.11s-x4isdvp	ghostscript/10.0.0-3atesdh	mpfr/4.2.0-
n2tkxso	rust/1.73.0-647r2tw	
bazel/6.1.1-vvtxktr	gimp/2.10.32-tlknk2n	mriconvert/2.1.0-
oaq24fz	rust/1.81.0-3qqoayc	(D)
bbmap/39.01-jnnkpwk	git-lfs/3.3.0-laphnvj	mricrogl/2022.07.20-
n3b7whc	sage-container/10.3-avpqipf	
bcftools/1.13-76jesdj	git/2.44.0-6f7n7ni	mricon/201909-
3s2phrj	sage/9.5-drpqjkh	
bcftools/1.16-ewu6fpe	(D) glew/2.2.0-plawm2j	msmc2/2.1.4-
cuac55f	sage/10.3-nntihfr	(D)
bcl2fastq2/2.20.0.422-z3wh636	glm/0.9.9.8-m3s6sze	multiwfn/3.8_1208-
qn65pif	salmon/1.9.0-itdua6n	
beagle/5.4-e43mqsa	glpk/5.0-zifs7bb	mummer4/4.0.0rc1-
4llgadq	salmon/1.10.1-43ljn7g	(D)
bedops/2.4.40-bjb2v2n	gmap-gsnap/2024-08-20-dur7jyc	muscle/3.8.1551-
pys2b76	samtools/1.12-4v4uiz6	
bedtools2/2.31.0-lsohc7s	gmp/6.2.1-qlaig4m	mysql/8.0.29-
w7xdbde	samtools/1.16.1-txuglks	(D)
bismark/0.23.0-eoksupu	gnuplot/5.4.3-pdiiquy	nanoflann/1.4.3-
u2l24dv	sas/9.4m8-f2f3xdp	

blast-legacy/2.2.26-tcdku3a	go/1.17.1-f4mqosa	nbo/7.0-
phausw3	schmutzi-container/1.5.7-vsorpcq	
blast-plus/2.2.30-cyxlprt	go/1.20.3-xknmcqd	nccl/2.16.2-1-
gjmprw5	schmutzi-container/1.5.7-3imwf7h (D)	
blat/37-ebfj5e6	go/1.23.3-d3wvs6z (D)	ncdu/1.18.1-
uofylp6	schrodinger/2023-4-h3kvbn3	
blender/4.4.0-446jdg	google-cloud-cli/456.0.0-3mtj4z6	nco/5.1.5-
36hru5t	schrodinger/2024-1-yqi27-m4qqm46 (D)	
boost/1.80.0-harukoy	gperf/3.1-56q4xf5	ncview/2.1.8-
nxepxtw	scons/4.5.2-housgyw	
bowtie/1.3.1-2kd7din	grace/5.1.25-duvo7rn	neovim/0.9.4-
67stov2	seqkit/0.10.1-qtiftw4	
bowtie2/2.4.2-xdquyzq	graphviz/8.0.1-75znavc	neovim/0.10.4-
7b4mer5 (D)	seqtk/1.3-kbdjwob	
bowtie2/2.5.3-qgsc2u (D)	gsl/2.7.1-khmyfcy	netcdf-c/4.9.2-
cfggqwi	shapeit4/4.2.2-3us45un	
brotli/1.0.9-h22dril	guppy/6.0.1-wpaqayj	netcdf-c/4.9.2-4ozokng
(D) singular/4.4.0-aeloppr		
bwa/0.7.17-lu4b4dj	guppy/6.1.2-wwwvdfu (D)	netcdf-cxx4/4.3.1-
6gcdg5s	skewer/0.2.2-nwhklgr	
bxh-xcede-tools-container/1.11.14-4sphv7n	gurobi/10.0.1-q7rc5dw	netcdf-fortran/4.6.0-
kl27oji	slicer/5.4.0-rb2kk4l	
cadence/IC06.18-calibre2022.2-ascb7dw	hdf5/1.12.2-s6aacp3	netcdf/4.9.2-
ar77jpt	slim/4.0.1-kymgtmu	
cadence/IC06.18.090-6famfci	hdf5/1.14.1-2-rdd6y6v (D)	netlib-lapack/3.11.0-
jdzmstx	slim/4.3-u22vcwu (D)	
cadence/IC23.10.000-ppql2n (D)	hisat2/2.2.1-gn4pb3l	netlogo/6.4.0-
psm765m	spdlog/1.11.0-qbi24my	
casa/6.6.0-20-py3.8-el7-dqvn5lw	homer/4.11.1-fpjs4l4	netpbm/10.73.43-
m2jdopk	splash/2.1.4-unrsfpj	
cdhit/4.8.1-bqmf4jf	hpcx-mpi/4.1.5rc2-mts-ukpby4i	ngc-jax/23.10-paxml-py3-
ziphcif	spm/8-77b5myx	
cellranger/arc-2.0.1-uamrhhu	hpcx-mpi/4.1.5rc2s-yflad4v (D)	ngc-pytorch/24.03-py3-
a6ptiby	spm/12_r7606-prcq7fg (D)	
cellranger/atac-2.0.0-m2tfcpk	htop/3.2.2-kqsjlaj	ngc-tensorflow/24.03-tf2-py3-
lmnuwwg	sratoolkit/3.0.0-u4jvgps	
cellranger/6.0.0-dbzt7r (D)	htslib/1.12-ecidzx4	ninja/1.11.1-
k2aq3rl	stacks/2.65-geg4r7a	
cfitsio/4.2.0-5grfqtu	htslib/1.17-zxcat2k (D)	nlopt/2.7.1-fwj27pk
star/2.7.10b-fj6kao2		

cgal/5.4.1-64mikh	idba/1.1.3-nrxiqtw	nnn/4.9-r7kawsv
stata/mp17-v7a7uoo		
chrome/119.0.6045.159s-avadhvk	idemp/201706-a45gc3d	node-js/18.12.1-
qkps4za	stata/mp18-3wq5b4o	(D)
cli11/2.3.2-pcucv7l	idl/8.9s-jocgnbh	nvhpc/23.3-xa4nyqi
stow/2.4.0-y2q7tsn		
clustal-omega/1.2.4-mbj3dq5	igraph/0.7.1-wbiepb3	nvhpc/23.7-
alnzdzw	stringtie/2.2.1-7uti3ny	
cmake/3.6.1-il7bkvj	imagej/154-linux64-java8-jd6sflr	nvhpc/25.1-
scvc2ac	(D) sublime-text/4.4143-im3loi3	
cmake/3.26.3-xi6h36u	(D) imagemagick/7.1.1-3-ex4k4u2	nvtop/3.0.1-
7r22pjl	subread/2.0.2-5agghnd	
cnvnator/0.4.1-w3bkqjf	inkscape/1.3s-gshcpwc	octopus-lunter/0.7.4-
kkqrfv3	swig/4.1.1-bq46cxl	
code-server/4.20.0-tcrmrsm	intel-oneapi-compilers/2023.1.0-a7fw7qt	ollama/0.3.14s-
a3gonhs	synopsys/2023.12-df4a3ab	
code-server/4.97.2-33bvsj5	(D) intel-oneapi-mkl/2023.1.0-xbcd2g3	ollama/0.5.12s-
mjarasi	(D) synopsys/2024.09-q3jwzot	(D)
colordiff/1.0.21-ifskyqr	intltool/0.51.0-vanhjsr	openbabel/3.1.1-
nay2mkb	tabix/2013-12-16-d6qvxp7	
comsol/5.2-ufifhtv	iq-tree/2.1.3-gu64b4j	openblas/0.3.23-
u6k5fey	tcl/8.6.12-dziqp2l	
comsol/5.6_yqi27-jnspqto	iq-tree/2.3.6-fn5vscb	(D) opencv/4.6.0s-
5z4piup	tcsh/6.24.10-dtqo5ky	
comsol/6.3_yqi27-7yf67lt	(D) iraf/2.17.1s-2r7ypc5	openexr/3.1.5-
6fapou6	tecplot/2022r1-q5cg2zq	
conn/22a-nztrdv3	itk-snap-container/4.0.2-bychj73	openjdk/11.0.17_8-
nw5ylvi	tesseract/4.1.1-l2ejycz	
connectome-workbench/1.5.0-t66riqu	jags/4.3.1-4mvaxc3	openjdk/17.0.5_8-
pq2e7ao	(D) tesseract/5.3.3-vq3altd	(D)
cppunit/1.14.0-h3hsjgu	jellyfish/2.2.7-dywzm7z	openjpeg/2.5.0-
iyu5vwb	texlive/20220321-pocclv	
crossrate-container/2016-27ofi4r	jo/1.9-ki7xc2u	openmpi/4.1.2-
s5wtoqb	texstudio/3.0.1-64vxo64	
cuda/10.1.243-bxisbai	json-fortran/8.3.0-ehkzpjv	openmpi/4.1.4s-
smqniuf	tk/8.6.11-uqpqlly	
cuda/10.2.89-xnfjmrtd	jsoncpp/1.9.5-vhsa2iy	openmpi/4.1.5-
kzuexje	tmux/3.3a-zyhjvvh	
cuda/11.8.0-lpttyok	julia/1.9.3s-i3zndt3	openmpi/5.0.1s-6ti4ij7
tmux/3.5a-kcjtgdnd	(D)	

cuda/12.1.1-ebglvvq	julia/1.11.1s-kueea5s	(D) openmpi/5.0.2s-
mfj2kfp	(D) tn93/1.0.12-tcvbyl4	
cuda/12.2.0-4lgnkrh	kraken/1.1.1-e6r2aej	openslide/3.4.1-
pzjb2kl	tree/2.1.0-7tlhzo7	
cuda/12.3.0-r72aozf	lemon/1.3.1-jh3h4xt	openssl/1.1.1t-
u2rkdf	trimal/1.4.1-ace7du2	
cuda/12.4.0-piq32fy	(D) leptonica/1.81.0-pebiyok	or-tools/9.10-
k4nov4d	trimgalore/0.6.6-iwfrq4c	
cudnn/7.5.1.10-10.1-hv4e2lt	leveldb/1.23-f76iwfr	ovito/3.6.0-
rile7ax	trimgalore/0.6.9-hisz5xp	(D)
cudnn/8.7.0.84-11.8-lg2dpd5	lftp/4.9.2-vimt4vf	p7zip/17.05-
3xtimiz	trimmomatic/0.39-w5jnhai	
cudnn/8.9.6.50-12-56zgdoa	libarchive/3.6.2-mnc5shn	pandoc/2.19.2-
wawlx5m	udunits/2.2.28-rycabdx	
cudnn/9.8.0.87-12-j5i4iki	(D) libbeef/Nov2020-xhrdwg5	pangolin/0.6-
vwij3iv	usearch/11.0.667-wx6utmj	
cufflinks/2.2.1-ogzw3z5	libdeflate/1.10-5yi7m3g	parallel/20220522-
5ah2i5h	v8/3.14.5-aompxje	
cutensor/1.5.0.3-gqkzath	libgd/2.2.4-2iyhgxa	paraview/5.9.0s-
dgv24kr	vasp-mpi/5.4.4-mdh3hpy	
datamash/1.8-ib4aakp	libgd/2.3.3-ubu4k2f	(D) patchelf/0.17.2-
aqmx4qb	vasp-mpi/5.4.4-wannier-cqhzfma	
dcm2niix/1.0.20220720-nwsidfo	libgeotiff/1.6.0-voueb6b	paup/4.0a168-
cwt24ux	vasp-mpi/6.3.2_avandewa-chn3w3j	
diamond/2.0.15-h7xx24l	libgit2/1.6.4-a432pgi	pcre2/10.42-
xks64jg	vasp-mpi/6.4.2_cfgoldsm_vtst-cff5qmk	
dicombrowser/20181217s-ikvqhry	libiconv/1.17-jwjcds2	pdftk/2.02-
gu7lpeg	vasp-mpi/6.4.2_cfgoldsm-7krhcss	
dlib/19.22-lxah7rq	libjpeg-turbo/2.1.5-sewtk5u	pdsh-chaos/23.12-
t6ywlrp	vasp-mpi/6.4.3_cfgoldsm-5dioee2	
dmtcp/3.0.0-xvfukfp	libjpeg/9e-6djp5nd	perl-dbi/1.643-
t74vmeb	vasp-mpi/6.4.3_yqi27-wannier-livyes6	
dorado/0.8.2-s42dhri	libnsl/1.3.0-calriiy	perl/5.26.2-o4iq4b4
vasp-mpi/6.4.3_yqi27-6uzdgnw	(D)	
dos2unix/7.4.2-5a6dlgt	libnsl/2.0.1-ed2i5hn	(D) perl/5.36.0-bt34quz
(D) vcftools/0.1.14-syssqsi		
dotnet/8.0.100-5lr7bga	libpng/1.2.57s-lve65hz	perl/5.37.9-
og4osvm	vim/9.1.0867-wl3haj7	
dropest/0.8.6-ewwx5ik	libpng/1.5.30-ru3zswz	perl/5.40.0-
o7hxcic2	virtualgl/3.1-yphbrfj	

ds/9.8.5s-zpqg2jy	libpng/1.6.39-ryxiwrd	(D)	picard/2.26.2-
qabtyqy	visit-container/3.3.3-qz3dni6		
dsi-studio/chen-2023-sif-lytwlk2	libreoffice/7.2.2.sif-drpjygp		pigz/2.7-
zgdIry3	visit-mpi/3.3.3s-lz2dp7m		
dtitk/2.3.1s-bp7yqjh	libsdl/2.30.7-4rxs72s		plink/1.9-beta6.27-
nvY4vrX	vmd/1.9.3-oin2dnj		
eigen/3.4.0-uyckkhi	libsodium/1.0.20-4o75gk3		plink/2.00-b6x44xw
(D)	vscode/1.84.2-4tfimgp		
eigensoft/7.2.1-6ctbhoz	libtiff/4.5.0-g6fga7e		popoolation2/1.205s-
luyjn2b	vscode/1.97.2-txhgk3l	(D)	
emacs/28.2-rwds2pd	libtree/3.1.1-yxst452		postgresql/16.4-
vzmexkn	wcstools/3.9.7-lo4forb		
expat/2.5.0-zujcztp	libvips/8.13.3-ex4pfpg		prodigal/2.6.3-
vbq7usx	wxwidgets/3.2.2.1-mk5eiyq		
fastme/2.1.5.1-kmg5til	libwnck/3.24.1-4gvjyhg		proj/9.2.0-
ni5rcfb	xcrysden/1.5.60-nuxe46i		
fastp/0.23.4-xmfbk37	libx11/1.8.4-qdzkebe		protobuf/3.22.2-
6hlkkut	xerces-c/3.3.0-djupwmt		
fastq-screen/0.15.3-7ymgrux	libxc/4.3.4-uy5ogwb		py-ase/3.21.0-
pyuljod	xeyes/1.2.0-nge56yb		
fastqc/0.11.9-mvd2uhw	libxc/5.2.3-ncc5ir4	(D)	py-matplotlib/3.7.1-
4afsjsz	xgboost/1.6.2-fp3ii65		
fastqc/0.12.1-sk2rb3a	(D)	libyaml/0.2.5-wdpYE7g	py-statsmodels/0.13.2-
sbdhj4k	yaml-cpp/0.7.0-6nno2ru		
fasttree/2.1.11-o5kvig7	libzip/1.3.2-qwqikw6		py-sympy/1.11.1-
gqgr7wu	zlib/1.2.13-jv5y5e7		
fastx-toolkit/0.0.14-zhaxiyn	liggghts/3.8.0s-rqph5mk		pycharm-community/2021.3.3-
weqrclY	zoxide/0.9.2-ydhigq6		
ferret/7.6.0-i6u5m7q	linaro-forge/23.1.2-pk2lobu		pypy/7.3.13-
6a5ma5j	zstd/1.5.5-zokfqsc		
ffmpeg/6.0-fy677gn	llvm/16.0.2-mq6g5lb		python/3.9.16s-x3wdtvt
ffmpeg/7.0-xny2fb2	(D)	lmod/8.7.24-w2akdkb	python/3.11.0s-
ixrhC3q	(D)		
fiji/20231107-1617-espdc7g	macaulay2-container/1.2-vupc5gy		qgis/3.28.3-5axmqsj

Oscar: Sage

Loading and Launching Sage

1. Once authenticated to Oscar, use the following commands at the command line.
2. Start an interactive job by using the `interact` command. This command can take additional parameters to extend the resources and time allotted to the node as well as the partition that the node operates on.
3. The Sage module provides containers. To load them, use `module load sage-container/10.3`.
4. To start the container, use `apptainer shell /oscar/rt/9.2/software/0.20-generic/0.20.1/opt/spack/linux-rhel9-x86_64_v3/gcc-11.3.1/sage-container-10.3-avpqi pfsnbneig726l72jrgdmlrivg4m/sage.sif`
5. Once inside the container's shell, use `sage` to launch the Sage console.

Sage on Oscar OnDemand

The easiest way to run Sage on Oscar OcDemand is to run sage in an interactive job via the terminal in your OnDemand session.

Use the `interact` command with parameters for your specific job to start the interactive session, then load your modules and run the sage binary (steps 2-4 above).

```
interact -n 2 -m 32g -t 04:00:00 -f 'haswell|broadwell|skylake'
```

Using Sage with Batch Scripts

Thanks to Trevor Hyde from Summer@ICERM 2019 for these instructions.

One method for running computations with Sage on Oscar is to write a script and use the slurm batch scheduler to have Oscar run your script. This requires two pieces:

1. A shell script to configure and submit your batch job to the cluster.
2. Your Sage code/program you'd like to run.

Example Batch Script

sage-batch.sh

```
#!/bin/bash

#SBATCH -J test_program
#SBATCH --array=0-9
#SBATCH -t 1:00:00
#SBATCH --mem=8G

#SBATCH -e data/<oscar-username>/test_output/test%a.err
#SBATCH -o data/<oscar-username>/test_output/test%a.out

module load sage-container/10.3

apptainer shell /oscar/rt/9.2/software/0.20-generic/0.20.1/opt/spack/linux-rhel9-x86_64_v3/gcc-11.3.1/sage-
container-10.3-avpqiipfsnbneig726l72jrgdmlrivg4m/sage.sif

sage test_program.sage $SLURM_ARRAY_TASK_ID
```

- `#!/bin/bash` tells the system this is a bash (shell) script.
- `#SBATCH -J test_program` sets the name of the job which appears when you check the status of your jobs.
- `#SBATCH --array=0-9` is an easy way of doing parallel computations. In this case it says our job will run on 10 different nodes, each node will be passed a parameter and we have specified that the parameters will take the values 0 through 9. You can specify several ranges or even list individual parameters if you prefer.
- `#SBATCH -t 1:00:00` specifies a time limit in `HH:MM:SS` for each node. Once this time runs out your program will stop running on that node. Be careful setting the time limit too high as doing so may make it take a long time for your job to get scheduled to run. Before starting a big computation try to do some smaller tests to see how long you expect to need.
- `#SBATCH --mem=8G` specifies how much memory each node gets. Standard exploratory accounts get 123GB total to use at any one time. So if you allocate too much per job, fewer jobs will run at once. On the other hand, if you allocate too little and a computation needs more than it has, then it will terminate. If this happens an “out of memory” error will show up in the `.err` file for that node.

- `#SBATCH -e data/<ccv-username>/test_output/test%a.err` and `#SBATCH -o data/<ccv-username>/test_output/test%a.out` specify where the error messages and output for each computation should be sent. You should store these files in your user folder, not on the submit node. We each have a folder inside the `data` directory which you can see from the submit node. In this example I have created a folder titled `test_output` where I'm putting both of these files. **You need to make these folders before you run the computation otherwise the output will be dumped into the void!** The `%a` will get replaced with the array parameter. So for example, since we set our array parameters to be `0-9` there will be 10 nodes running and each of them gets a number between 0 and 9; this node corresponding to the parameter 7 will create two files `test7.err` and `test7.out`.
- `module load sage-container/10.3` loads the sage container into the node.
- `apptainer shell /oscar/rt/9.2/software/0.20-generic/0.20.1/opt/spack/linux-rhel9-x86_64_v3/gcc-11.3.1/sage-container-10.3-avpqi pfsnbneig726l72jrgdmlrivg4m/sage.sif` initiates the container's Sage console shell.

Everything after this in the script happens as if you typed it yourself onto the command line.

- In our example, we want to run sage code, so the line `sage test_program.sage` `$SLURM_ARRAY_TASK_ID` runs our example sage program `test_program.sage`.
- The file needs to have the `.sage` extension.
- You should write this file in a text editor, not in a Jupyter notebook (although you can first write and test your program in a Jupyter notebook and then copy and paste it into a new file when it's ready).
- This program is written to accept one input and I have passed it `$SLURM_ARRAY_TASK_ID` which is the array parameter passed to each node. You can use this parameter to select which input parameters to run your program on.

Example Sage Program

`test_program.sage`

```
import sys

def fun_math(message):
    print message
    sys.stdout.flush()

job_id = int(sys.argv[1])
fun_math('hi this is a test')
fun_math('my job id is' + str(job_id))
```

- In the Sage program, you first define all of your functions and then you include the code you want to run.
- Import `sys` so you can access the array parameter passed to your function from the node. This is accessed in this case by `sys.argv[1]`. Make sure you explicitly coerce to be an integer if you want to use it as an integer; it's a string by default.
- The output of the `print` command is appended to the `.out` file for this node as a new line.
- Notice the line `sys.stdout.flush()` included in the function. This makes the program immediately send whatever output it has to the output file when called. Otherwise the program won't output **anything** until it has completely finished running. If each node is running 100 potentially long computations and it finishes the first 99 but then times out on the 100th computation, and you don't include any `sys.stdout.flush()` commands, everything will be lost when time runs out.

Submitting the Batch Job

- To run this batch program go back to the submit node and type `sbatch <NAME_OF_BATCH_FILE>`. In our example here, our batch file is called `sage-batch.sh`, so we simply type `sbatch sage-batch.sh`. Slurm will return a line that tells you your job has been submitted together with a job id number.
- To check the progress of your jobs type `myq` from anywhere on Oscar. This will show you what jobs you have running, how much time they have left, and which jobs are still waiting to run. Be patient, sometimes it takes a minute for things to get started.
- If you realize your code is never going to finish or that you've made some terrible mistake, you can cancel a batch job by typing `scancel <JOB_ID>`. You can specify a single node or just put the general job id for the whole run and cancel everything.

Oscar: MATLAB

Loading and Launching MATLAB

1. Open the Terminal and use the following commands at the command line.
2. `module avail matlab` to list all the available matlab versions.
3. `module load matlab` to load the latest version of matlab (R2023a). Other versions can be specified with the command `module load matlab/R2019a`.
4. `matlab` to launch the MATLAB app.

Installing MATLAB Packages such as YALMIP

MATLAB script packages, such as YALMIP, can be installed directly by the user on their Oscar account.

1. Open the Terminal and connect to Oscar.
2. Navigate to your home folder by typing `cd ~`
3. `mkdir -p MATLAB`
4. `wget -O yalmip.zip https://github.com/yalmip/yalmip/archive/master.zip`
5. `unzip yalmip.zip`
6. In MATLAB, add the YALMIP-master directory to your path.
 1. In the MATLAB file browser, navigate to the MATLAB folder you created in your home folder. `cd ~/MATLAB`
 2. Right click on the YALMIP-master folder.

3. Select Add to Path > Selected Folders and Subfolders. This adds the YALMIP folders to your path.

7. To save your MATLAB path, use the savepath command in the MATLAB command prompt.

```
savepath ~/MATLAB/pathdef.m
```

YALMIP also requires a solver like SDPT3. The steps below add SDPT3 to MATLAB.

1. Open the Terminal.

2. `cd ~/MATLAB`

3. `wget -O sdpt3.zip https://github.com/sqlp/sdpt3/archive/master.zip`

4. `unzip sdpt3.zip`

5. In MATLAB, add the sdpt3 directory to your path.

1. In the MATLAB file browser, navigate to the MATLAB folder you created in your home folder. `cd ~/MATLAB`

2. Right click on the sdpt3-master folder.

3. Select Add to Path > Selected Folders and Subfolders. This adds the SDPT3 folders to your path.

6. To update/save your MATLAB path, use the savepath command in the MATLAB command prompt. `savepath ~/MATLAB/pathdef.m`

Oscar: Mathematica

Loading and Launching Mathematica

1. Open the Terminal and use the following commands at the command line.
2. Start an interactive job by using the interact command. This command can take additional parameters to extend the resources and time allotted to the node as well as the partition that the node operates on.
3. Command `module avail mathematica` to list all the available Mathematica versions.
4. Command `module load mathematica` to load the latest Mathematica version.
5. Command `mathematica` to launch the Mathematica app.

Oscar: R

Loading and Launching R

1. Once authenticated to Oscar, use the following commands at the command line.
2. Start an interactive job by using the `interact` command. This command can take additional parameters to extend the resources and time allotted to the node as well as the partition that the node operates on.
3. To load R, use `module load r/4.4.2-re5rjx3`, or another version of your preference.
4. Once inside the container's shell, use `R` to launch the R console.

Installing R packages

1. R packages need to be installed locally.
2. First, load the R version that you want to use the package with: `module load r/4.2.2`
3. Start R Session `R`
4. For packages that require code to be compiled, install them in the log in node. Refer to more information [here](#)
5. To reinstall packages, start a new session by:

```
module load r/4.2.2  
R
```

6. Then, update packages with `update.packages(checkBuilt=TRUE, ask=FALSE)`
7. To uninstall packages, start an R session. Then run `remove.packages("wordcloud")` (word cloud can be replaced with string of another package name)

Oscar: Macaulay 2

Loading and launching Macaulay 2

Once authenticated to Oscar, use the following commands at the command line.

1. Start an interactive job by using the `interact` command. This command can take additional parameters to extend the resources and time allotted to the node as well as the partition that the node operates on.
2. The Macaulay 2 module provides containers. To load them, use `module load macaulay2-container/1.2-vupc5g`
3. To start the container use `apptainer shell $MACAULAY2_CONTAINER`
4. Once inside the container's shell, use `M2` to launch the Macaulay console.

Oscar: Magma

Loading and launching Magma

1. Once authenticated to Oscar, use the following commands at the command line.
2. Magma requires a GPU partition. Start an interactive node by commanding `interact -q gpu"`.
3. To load Magma, use `module load magma-usr/V2.28-8-p3zylpg`.
4. Use `magma` to launch the Magma console.