2025/10/29 04:40 1/33 Applications on the clusters

# **Applications and libraries**

```
Table of Contents
  Applications on the clusters
         Module - Imod
               How to use 'module'
                What do I do when an application is not available via 'module'?
         Detailed example of using 'module'
                Loading 'R'
         Choosing the compiler toolchain
                FOSS toolchain
                Intel toolchain
                      Intel compiler licenses
                fosscuda toolchain
  Examples for selected applications
         OpenMPI
                Specify MCA parameters through "srun"
         Conda
                How to Create a Conda Environment in a Container
                      Benefits
                      Limitations
                      Step 1 - Define the Conda Environment
                      Step 2 - Build the Container
                      Step 3 - Use the Container
                Conda environment management
                Package management
         ADF
         Gaussian
         Git
         NVIDIA HPC SDK Installation in Home Directory on a Heterogeneous Cluster
                Prerequisites
                Installation Steps
                Verify Installation
         Gurobi
         Jupyter notebook and Jupyter lab
         Mathematica
         Matlab
                Parallel with Matlab
                Pass sbatch arguments to Matlab
                Compile your Matlab code
                Matlab PATH
                Matlab java.opts
                CHROMIUM mailbox/texture errors
                Wavelab
         OpenCL
         Distant Paraview
         Python
                Custom Python lib
                Pip install from source
         R project and RStudio
                RStudio
                R packages
         Variant Effect Predictor (VEP)
                Install species
         Apptainer (was Singularity)
                Intro
```

Last update: 2025/06/11 12:27

> Pull an existing image Convert a Docker image Run a container Modify the image (not persistent) Modify the image (persistent) References

Stata

TensorFlow

Compile and install a software in your /home

# **Applications on the clusters**

An important number of applications and libraries are available on Baobab and Yggdrasil, and we often offer multiple versions.

The magic which allows that is lmod with the module command. This is the recommended way to load any application or libraries on the clusters.

On this page, we will give the most common module usage and give some example for a selection of applications.

# **Module - Imod**

The recommended way to load an application on the clusters is to use the module command. By using module, you don't need to know where the software and libraries are physically located (the full path), but instead you can just type the application name as if its path was in your PATH (this is indeed what module does).

The module command can also set other important environment variables for an application, so it is always recommended to use it.

#### How to use 'module'

These are the most common options you will use with module.

To get a complete list of applications available with module

module spider

To find the available versions of a certain application :

module spider <app\_name>

To load one (or more) application:

module load <app\_name\_1> <app\_name\_2> ...

2025/10/29 04:40 3/33 Applications on the clusters

To load a specific version of an application :

```
module load <app name/version>
```

**Hint**: Module version. By choosing a module without specifying a version, you will always get the latest version available. However, we always recommend to specify the version, as your code might produce different results if you are using another version. If **reproducibility of results** is important for you, you should definitely used a fixed version.

You can see the help for a particular module (it must be loaded first):

module help R

See the list of currently loaded modules :

module list

To unload all currently loaded modules:

module purge

**Hint**: If you are in a hurry you can also use ml instead of module for any of the above mentioned commands:

ml spider

For more information, use the manpage man module.

# What do I do when an application is not available via 'module'?

If the application you need or the exact version is not available via module:

- First drop us an email at hpc@unige.ch and explain with as much detail as possible what you need (provide scripts, links, etc.). If we can install what you need, we will do it, most of the time via EasyBuild/module (you can check the list of available software).
- If we cannot, you can compile binaries in your \$H0ME directory and use them on any node of the cluster (since your \$H0ME is accessible from any node). Make sure you load the compiler with module first!
  - Read more in the section Compile and install a software in your /home
- Another interesting option is to use Singularity which allows you to run containers on the clusters.

# **Detailed example of using 'module'**

# Loading 'R'

Let's go through an example of loading R.

First, let's find all available version of R:

```
[brero@login2 ~]$ module spider R
_____
   Description:
     R is a free software environment for statistical computing and
graphics.
    Versions:
       R/3.2.3
       R/3.3.1
       R/3.3.2
       R/3.5.1
       R/3.6.0
       R/3.6.2
       R/4.0.0
    Other possible modules matches:
       APR APR-util
                    BioPerl
                             Bismark Blender CellProfiler CellRanger
CoordgenLibs DISCOVARdenovo
                           DendroPy ...
[\ldots]
```

Now some people just need the latest version available and can simply load it with module load R; whitout specifying a version, you will always get the latest version.

But sometimes, you need to use the same version everytime and we recommend it. This is very important as your code might produce different results if you are using another version. If reproducibility of results is important for you, you should definitely used a fixed version.

To load a specific version, in this case R version 3.6.2:

```
[brero@login2 ~]$ module load R/3.6.2

Lmod has detected the following error: These module(s) or extension(s)

exist but cannot be loaded as requested:

"R/3.6.2"

Try: "module spider R/3.6.2" to see how to load the module(s).
```

This fails as the module "cannot be loaded as requested". This is usually because you are missing dependencies.

The message also suggest to try the following command:

[brero@login2 ~]\$ module spider R/3.6.2
R: R/3.6.2

2025/10/29 04:40 5/33 Applications on the clusters

```
Description:
      R is a free software environment for statistical computing and
graphics.
   You will need to load all module(s) on any one of the lines below before
the "R/3.6.2" module is available to load.
     GCC/8.3.0 OpenMPI/3.1.4
   Help:
      Description
      R is a free software environment for statistical computing
       and graphics.
     More information
       - Homepage: https://www.r-project.org/
      Included extensions
      ============
      abc-2.1, abc.data-1.0, abe-3.0.1, abind-1.4-5, acepack-1.4.1,
adabag-4.2,
      ade4-1.7-13, ADGofTest-0.3, aggregation-1.0.1, akima-0.6-2,
AlgDesign-1.2.0,
      animation-2.6, aod-1.3.1, ape-5.3, arm-1.10-1, askpass-1.1,
asnipe-1.1.12,
      assertthat-0.2.1, AUC-0.3.0, audio-0.1-6, b-a, backports-1.1.5,
bacr-1.0.1,
      bartMachine-1.2.4.2, bartMachineJARs-1.1, base64-2.0, base64enc-0.1-3,
      BatchJobs-1.8, BayesianTools-0.1.7, bayesm-3.1-4, BayesPen-1.0,
BB-2019.10-1,
      BBmisc-1.11, bbmle-1.0.20, BCEE-1.2, BDgraph-2.62, bdsmatrix-1.3-3,
      beanplot-1.2, beeswarm-0.2.3, BH-1.69.0-1, BiasedUrn-1.07,
bibtex-0.4.2,
      bigmemory-4.5.33, bigmemory.sri-0.1.3, bindr-0.1.1, bindrcpp-0.2.2,
      bio3d-2.4-0, biom-0.3.12, bit-1.1-14, bit64-0.9-7, bitops-1.0-6,
blob-1.2.0
[...]
```

As the message explains, you **need** to load 2 dependencies GCC/8.3.0 and OpenMPI/3.1.4 before you can load R.

You can then simply execute the following command:

```
[brero@login2 ~]$ module load GCC/8.3.0 OpenMPI/3.1.4 R/3.6.2
```

Remember that GCC and OpenMPI are packaged together in the foss module. You can then load the corresponding foss module instead which is shorter:

```
[brero@login2 ~]$ module load foss/2019b R/3.6.2
```

Then you can just invoke R by typing R in the terminal (instead of using the full path). Of course you are still required to use Slurm and an sbatch script to launch your software.

**Hint**: To automatically load some modules at login, you can add something like this in your \$HOME/.bashrc:

```
if [ -z "$BASHRC_READ" ]; then
  export BASHRC_READ=1
  # Place any module commands here
  module load GCC/8.3.0 OpenMPI/3.1.4 R/3.6.2
fi
```

# Choosing the compiler toolchain

You have the choice between FOSS toolchain or Intel toolchain (license required).

If you want to compile your software against MPI, it is very important not to compile using directly gcc, icc or similar commands, but rather rely on the wrappers mpicc, mpic++, mpicxx or similar ones provided by module.

All the newer versions of MPI will be available through the use of module.

#### **FOSS toolchain**

module	compiler	mpi
foss/2016a	gcc 4.9.3	openmpi 1.10.2
foss/2016b	gcc 5.4.0	openmpi 1.10.3
foss/2017a	gcc 6.3.0	openmpi 2.0.2
foss/2017b	gcc 6.4.0	openmpi 2.1.1
foss/2018a	gcc 6.4.0	openmpi 2.1.2
foss/2018b	gcc 7.3.0	openmpi 3.1.1
foss/2019a	gcc 8.2.0	openmpi 3.1.3
foss/2019b	gcc 8.3.0	openmpi 3.1.4
foss/2020a	gcc 9.3.0	openmpi 4.0.3
foss/2020b	gcc 10.2.0	openmpi 4.0.5
foss/2021a	gcc 10.3.0	openmpi 4.1.1
foss/2021b	gcc 11.2.0	openmpi 4.1.1
foss/2022a	gcc 11.3.0	openmpi 4.1.4
foss/2022b	gcc 12.2.0	openmpi 4.1.4

2025/10/29 04:40 7/33 Applications on the clusters

module	compiler	mpi
foss/2023a	gcc 12.3.0	openmpi 4.1.5
foss/2023b	gcc 13.2.0	openmpi 4.1.6
foss/2024a	gcc 13.3.0	openmpi 5.0.3

Example for the latest version of gcc:

module load foss

If needed, you can stick to a particular (or legacy) version:

module load foss/2021b

You can see the details of what is loaded in foss with:

module list

## Intel toolchain

module	compiler	mpi
intel/2016a	icc 16.0.1	impi 5.1.2
intel/2016b	icc 16.0.3	impi 5.1.3
intel/2017a	icc 17.0.1	impi 2017 Update 1
intel/2018a	icc 18.0.1	impi 2018.1.163
intel/2019a	icc 19.0.1	impi 2018.4.274
intel/2020a	icc 19.1.1.217	impi 2019.7.217
intel/2021a	icc 2021.2.0	impi/2021.2.0
intel/2021b	icc 2021.4.0	impi/2021.4.0
intel/2022a	icc 2022.1.0	impi/2021.6.0

Example for the latest version of icc from Intel:

module load intel

If needed, you can stick to a particular (or legacy) version:

module load intel/2021a

You can see the details of what is loaded in intel with:

module list

If you want to use the intel compiler for mpi job, you need to export the variable:

export I\_MPI\_PMI\_LIBRARY=/usr/lib64/libpmi.so

intel mpi with slurm

#### Intel compiler licenses

If you want to use an **old** Intel compiler (before 2021a), you need to have your own Intel license compiler.

Once you get the license you should copy it to your home directory in a directory named Licenses (\$HOME/Licenses).



It isn't anymore possible to get this Intel license. Instead you are required to use Intel oneAPI which doesn't requires a license. Use intel/2021a or newer.

#### fosscuda toolchain

Cuda is provided through fosscuda or directly through CUDA module. The difference is that fosscuda is a bundle of software and CUDA only provide the minimum.

module	CUDA	GCC	OpenMPI
fosscuda/2018b	9.2.88	7.3.0	3.1.1
fosscuda/2019a	10.1.105	8.2.0	3.1.3
fosscuda/2019b	10.1.243	8.3.0	3.1.4
fosscuda/2020a	11.0.2	9.3.0	4.0.3
fosscuda/2020b	11.1.1	10.2.0	4.0.5

# **Examples for selected applications**

We are providing a bunch of sbatch examples on our GitLab repository here. Feel free to clone the repository and provide other examples or fixes through pull request.

# **OpenMPI**

# Specify MCA parameters through "srun"

Normally you would pass those parameters to `mpirun` but as you are using `srun` with Slurm, this is not possible directly.

You should have a dedicated file with parameters located here: \$HOME/.openmpi/mca-params.conf.

Or you may use environment variable with OMPI\_MCA\_ prefix. here for more details.

2025/10/29 04:40 9/33 Applications on the clusters

# Conda

#### How to Create a Conda Environment in a Container

Using **Conda** directly on HPC systems or shared servers can cause performance issues and storage overload because Conda environments create thousands of small files. This often results in:

- Slow job startup times
- Filesystem limitations being hit
- High I/O load on the cluster
- Complex environment management

A better solution is to **encapsulate Conda environments inside a container**. This way, the entire environment is packaged into a single file (such as a `.sif` image used by Apptainer/Singularity), which can be easily deployed, shared, and reused without polluting clusters.

#### **Benefits**

Using this method offers multiple advantages:

- 1. [] **Fewer files**: Your environment is stored in a single `.sif` file
- 2. [ Portability: Share the container easily with collaborators
- 3. [ Reproducibility: Your environment stays consistent across systems
- 4. ☐ **Isolation**: No risk of interfering with other environment
- 5. 

  Stability: Cluster updates will not break your environment

#### Limitations

1. △ The container is static; to update packages, you need to rebuild the image

This guide explains how to build such a container using cotainr, a tool that simplifies container creation.

#### **Step 1 - Define the Conda Environment**

Create a file env.yml that contains the definition of your environment: (As exemple we will use bioenv.yml)

name: bioenv
channels:

- bioconda
- conda-forge
- defaults

#### dependencies:

- blast=2.16.0
- diamond=2.1.11

```
- exonerate=2.4.0
- spades=4.1.0
- mafft=7.525
- trimal=1.5.0
- numpy
- joblib
- scipy
[...]
prefix:/home/users/a/alberta/env # <==== To delete/edit</pre>
```

You can generate this file using the following commands:

```
# 1. (optional) create your environment (or not if you already have one)
$ conda create -n bioenv -c bioconda -c conda-forge spades exonerate diamond
blast mafft trimal numpy joblib scipy -y

# 2. Activate your environment
$ conda activate bioenv

# 3. Export the settings of your environment
# It's recommended to manually remove the `prefix:` line at the bottom of
the file before using it with cotainr.
$ conda env export > bioenv.yml
```

#### Step 2 - Build the Container

Now use cotainr to create the image:

```
$ module load GCCcore/13.3.0 cotainr
# Ex: cotainr build <env.sif> --base-image=docker://<WhatEverYouWant>:latest
   --accept-licenses --conda-env=<env.yml>
$ cotainr build bioenv.sif --base-image=docker://ubuntu:latest --accept-licenses --conda-env=bioenv.yml
```

You can replace ubuntu: latest with any other base image, such as rockylinux: latest.

#### **Step 3 - Use the Container**

You can now run commands inside the container as follows:

```
$ apptainer exec bioenv.sif python3 -c "import numpy;
print(numpy.__version__)"
```

Or launch any program inside the container just like you would in a normal environment.

2025/10/29 04:40 11/33 Applications on the clusters

# **Conda environment management**

Use it

module load Anaconda3

Create

conda create --name environment\_name

As the conda env itself contains a lot of files, it may be a good idea to store it on the scratch folder for example.

conda create --prefix \$HOME/scratch/test-env --name environment name

List

conda env list

Activate

conda activate <env name>

Deactivate

conda deactivate

# Package management

List packages in a given environment:

conda list

# **ADF**

You can use SCM ADF on one ore more nodes of the cluster. Please note that you must use module (see Module - Imod) in you sbatch script to set the variables correctly.

Please see ADF example on GitLab for some example and scripts related to ADF.



Do not launch ADF using srun. ADF is a wrapper which uses srun internally.



ADF needs a fast local scratch space. On Baobab, the local scratch of each node is only about 180GB. If you need more space, you need to find another solution (do the



calculation on more nodes, do not use local scratch, buy us new hard disks)

By default, the local scratch space is defined in the SCM\_TMPDIR environment variable, /scratch as set by module load ADF. This default value will gives error when calling ADFview on the login nodes, given that /scratch does not exist there. You can overcome it by using the Linux default /tmp:

```
capello@login2:~$ module load ADF/2019.104
capello@login2:~$ SCM_TMPDIR=/tmp adfview
```

# Gaussian

You can use Gaussian g09 on one node of the cluster. Please note that you must use module (see Module - Imod) in you sbatch script to set the variables correctly. When using the module, it will set the variable GAUSS\_SCRDIR to /scratch of the local hard disk of the allocated node. This should lower the calculation time and as well lower the usage of the shared filesystem. See below for other optimizations.

There are two versions of g09 on the cluster. Revision c01 and d01.

Please see Gaussian example on GitLab for some examples and scripts related to Gaussian.

To optimize the run, you can add some lines in your job file. If you need more than 190 GB of scratch space, you should add the line (adapt /home/yourusername to your own path):

```
%RWF=/scratch/,170GB,/home/yourusername/scratch/,-1
```

You may as well specify how much memory you want to use. By default, Gaussian will use 250MB of ram. You can try with 50GB for example:

```
%Mem=50GB
```

You need to specify as well how many CPU cores you want to use:

```
%NProcShared=16
```

# Git

To use git on the cluster you need to do the following:

Add that to your \${HOME}/.gitconfig:

```
[core]
createObject = rename
```

To invoke git:

2025/10/29 04:40 13/33 Applications on the clusters

git clone --no-hardlinks

# **NVIDIA HPC SDK Installation in Home Directory on a Heterogeneous Cluster**

https://developer.nvidia.com/hpc-sdk

## **Prerequisites**

- Download and extract ONLY the tarball by following the procedure: https://developer.nvidia.com/hpc-sdk-downloads
- Ensure you have the latest version of GCC loaded. You can check and load the module using the following commands:

```
(base) (yggdrasil)-[alberta@login1 ~]$ ml GCC
(base) (yggdrasil)-[alberta@login1 ~]$ ml

Currently Loaded Modules:

GCCcore/13.2.0 2) zlib/1.2.13 3) binutils/2.40 4) GCC/13.2.0
```

# **Installation Steps**

#### 1. Start the NVIDIA HPC SDK Installer:

As it'writte press **enter** to continue

```
(base) (yggdrasil)-[alberta@login1 ~]$
nvhpc_2024_245_Linux_x86_64_cuda_12.4/install

Welcome to the NVIDIA HPC SDK Linux installer!

You are installing NVIDIA HPC SDK 2024 version 24.5 for Linux_x86_64.
Please note that all Trademarks and Marks are the properties
of their respective owners.

Press enter to continue...
```

2. **Select the Installation config:** Since we are setting up for a **heterogeneous** cluster, select the **network installation** option:

A single system installation is appropriate for a single system or a homogeneous cluster. A network installation should be selected for a heterogeneous cluster. For either a single system or network installation, the HPC SDK configuration (localrc) is created at install time and saved in the installation directory.

An auto installation is appropriate for any scenario. The HPC SDK configuration (localrc) is created at first use and stored in each user's home directory.

- 1 Single system install
- 2 Network install
- 3 Auto install

Please choose install option:

## 3. Specify the full path you want to install the software:

Please specify the directory path under which the software will be installed.

The default directory is /opt/nvidia/hpc\_sdk, but you may install anywhere you wish, assuming you have permission to do so.

Installation directory? [/opt/nvidia/hpc\_sdk]
/home/users/a/alberta/HPC\_sdk

Installing NVIDIA HPC SDK version 24.5 into /home/users/a/alberta/HPC\_sdk Making symbolic link in /home/users/a/alberta/HPC\_sdk/Linux\_x86\_64

generating environment modules for NV HPC SDK 24.5 ... done. Installation complete.

Please run add network host to create host specific localrc files:

/home/users/a/alberta/HPC\_sdk/Linux\_x86\_64/24.5/compilers/bin/localrc.\$host

on all other hosts you wish to run NVIDIA HPC SDK compilers.

For 64-bit NVIDIA HPC SDK compilers on 64-bit Linux systems, do the following:

/home/users/a/alberta/HPC\_sdk/Linux\_x86\_64/24.5/compilers/bin/add\_network\_host

HPC SDK successfully installed into /home/users/a/alberta/HPC sdk

If you use the Environment Modules package, that is, the module load command, the NVIDIA HPC SDK includes a script to set up the appropriate module files.

% module load /home/users/a/alberta/HPC\_sdk/modulefiles/nvhpc/24.5
% module load nvhpc/24.5

Alternatively, the shell environment may be initialized to use the HPC SDK.

In csh, use these commands:

2025/10/29 04:40 15/33 Applications on the clusters

```
% set path = (/home/users/a/alberta/HPC sdk/Linux x86 64/24.5/compilers/bin
$path)
% setenv MANPATH
/home/users/a/alberta/HPC_sdk/Linux_x86_64/24.5/compilers/man:"$MANPATH"
To use MPI, also set:
% set path =
(/home/users/a/alberta/HPC sdk/Linux x86 64/24.5/comm libs/mpi/bin $path)
In bash, sh, or ksh, use these commands:
$ export
PATH=/home/users/a/alberta/HPC_sdk/Linux_x86_64/24.5/compilers/bin:$PATH
$ export
MANPATH=/home/users/a/alberta/HPC sdk/Linux x86 64/24.5/compilers/man:$MANPA
TH
To use MPI, also set:
$ export
PATH=/home/users/a/alberta/HPC sdk/Linux x86 64/24.5/comm libs/mpi/bin:$PATH
Please check https://developer.nvidia.com for documentation,
use of NVIDIA HPC SDK software, and other questions.
```

Now the installation is complete **BUT** the module method does not work and you **MUST** use the export command.



If you haven't read the installation instructions carefully, do so now. Missing steps could lead to unforeseen adventures in troubleshooting! (An angry admin may RTFM you)

# **Verify Installation**

#### Create and compile a Test program

```
(yggdrasil)-[alberta@login1 ~]$ cat test.cpp
#include <ranges>
int main(){
  return 0;
}
(yggdrasil)-[alberta@login1 ~]$ cat sbatch_nv
#!/bin/bash
```

```
#SBATCH --job-name=compile test
#SBATCH --output=compile test.out
#SBATCH --error=compile test.err
#SBATCH --time=00:01:00
#SBATCH --partition=debug-cpu
# Load the necessary modules
ml GCC
# Set up the environment for NVIDIA HPC SDK
export HPC SDK DIR=/home/users/a/alberta/HPC sdk
export PATH=$HPC SDK DIR/Linux x86 64/24.5/compilers/bin:$PATH
export MANPATH=$HPC SDK DIR/Linux x86 64/24.5/compilers/man:$MANPATH
export PATH=$HPC SDK DIR/Linux x86 64/24.5/comm libs/mpi/bin:$PATH
# Run add network host to create host-specific localrc files
$HPC SDK DIR/Linux x86 64/24.5/compilers/bin/add network host
# Compile the test.cpp file
nvc++ test.cpp -o test
```

(yggdrasil)-[alberta@login1 ~]\$ sbatch sbatch\_nv Submitted batch job 33866816



- Ensure to run the add\_network\_host command on all nodes in your heterogeneous cluster to create the necessary localrc files.
- Make sure the environment variables are set in your shell configuration files (e.g., `.bashrc`, `.cshrc`) for persistent settings across sessions.
- For more details and further documentation, visit the [NVIDIA HPC SDK website](https://developer.nvidia.com).

# Gurobi

**NB** , the following instructions come from <a href="https://hpc-community.unige.ch/t/gurobi-solver-license-issue/459">https://hpc-community.unige.ch/t/gurobi-solver-license-issue/459</a>.

We provide an internal Gurobi token server, here how to use it:

```
capello@login2:~$ module load Gurobi
capello@login2:~$ gurobi_cl --tokens
Checking status of Gurobi token server 'master.cluster'...
```

2025/10/29 04:40 17/33 Applications on the clusters

```
Token server functioning normally.
Maximum allowed uses: 4096, current: 0
capello@login2:~$
```

# Jupyter notebook and Jupyter lab

Jupyter notebook can run on our clusters, however we do not have a dedicated server for Jupyter. This means you need to submit a job and request some resources before you can connect to your instance of Jupyter notebook.

The easiest way is to launch a session (on Baobab only for now) using OpenOnDemand

If it isn't working for you, check our git repo which as some example scripts to launch Jupyter as Slurm job.

You can also read: [tutorial] Jupyter notebook

or

updated Tutorial done for HPC-Lunch



For interactive programs such as Jupyter notebook, you might want to use the public-interactive-cpu partition (Which partition for my job)

# **Mathematica**

If you would like to use a different license server (by default mathlm.unige.ch), you can specify its URL in the \${HOME}/.Mathematica/Licensing/mathpass file, prependend by an exclamation mark:

```
capello@login2:~$ MATHEMATICA_LICENSE_SERVER_URL=mathlm.unige.ch
capello@login2:~$ mkdir -p ~/.Mathematica/Licensing/
capello@login2:~$ cat <<EOF >~/.Mathematica/Licensing/mathpass
!${MATHEMATICA_LICENSE_SERVER_URL}
EOF
capello@login2:~$ cat ${HOME}/.Mathematica/Licensing/mathpass
!mathlm.unige.ch
capello@login2:~$
```

See http://support.wolfram.com/kb/25655 and http://support.wolfram.com/kb/112 for more information.

# **Matlab**

Matlab is available in Baobab in different versions:

```
$ module spider matlab
-----
matlab:
----
Versions:
    MATLAB/2021a
    MATLAB/2021b
    MATLAB/2022a
```

Keep in mind that it's a licensed program, and that the licenses are shared with the whole university. To be fair with the other users, we have set up a limitation on the number of licenses you can use. We kindly ask you to specify in your sbatch file that you are using Matlab in order to keep the limitation effective. If you are using some licensed toolbox, like Wavelet\_Toolbox, you need to specify it as well. If you don't, your job may be killed without further notice in case we are out of licenses.

Example to specify that you need Matlab:

```
#SBATCH --licenses=matlab@matlablm.unige.ch
```

Example to specify that you need the Wavelet Toolbox:

```
#SBATCH --licenses=wavelet-toolbox@matlablm.unige.ch
```

See the licenses available on Baobab:

```
scontrol show lic
```

If you need a license not listed here, please ask us at hpc [at] unige [dot] ch.



You need to specify at LEAST the Matlab license, and zero or more toolbox.

To run Matlab in batch mode, you can create a batch file like this one:

```
#!/bin/bash

#SBATCH --cpus-per-task=1
#SBATCH --ntasks=1
#SBATCH --licenses=matlab@matlablm.unige.ch

module load MATLAB/2021b
```

2025/10/29 04:40 19/33 Applications on the clusters

```
BASE_MFILE_NAME=hello
unset DISPLAY
echo "Running ${BASE_MFILE_NAME}.m on $(hostname)"
srun matlab -nodesktop -nosplash -nodisplay -r ${BASE_MFILE_NAME}
```

In this example, you need to have your code in the file hello.m.

You submit the Matlab job like a normal sbatch SLURM job:

```
sbatch ./yourBatch
```

#### Parallel with Matlab

Since version 2014, you are not limited to 12 CPU cores anymore.

Please see Matlab on gitlab for some examples and scripts related to Matlab parallel

As we are talking about parallel and not distributed Matlab, you can consided Matlab as a multithread application.

See how to submit Multithreaded jobs.



If you are a parallel or distributed Matlab specialist and you have some hints, you are very welcome to contact us!

# Pass sbatch arguments to Matlab

You can pass arguments from sbatch to Matlab as described below.

Example of sbatch file (sbatch.sh)

```
[sbatch part as usual]

BASE_MFILE_NAME=test
MATLAB_MFILE=${BASE_MFILE_NAME}.m

unset DISPLAY

module load MATLAB/2021b

#the variable you want to pass to matlab
job_array_index=${SLURM_ARRAY_TASK_ID}
```

```
echo "Starting at $(date)"
echo "Running ${MATLAB_MFILE} on $(hostname)"

# we call the matlab function (see the parenthesis around the argument) and the argument type will be integer.
srun matlab -nodesktop -nosplash -nodisplay -r
"${BASE_MFILE_NAME}($job_array_index)"
echo "Finished at $(date)"
```

Example of Matlab file (test.m)

```
function test(job_array_index)
fprintf('array index: %d\n', job_array_index)
```

See arguments with Matlab on gitlab for more examples.

# **Compile your Matlab code**

Thanks to Philippe Esling for his contribution to this procedure.

The idea of compiling Matlab code is to save on licenses usage. Indeed, once compiled, a Matlab code can be run without using any license.

The Matlab compiler is named MCC

First load the needed modules:

```
module load foss/2016a matlab/2016b
```

Let's say you want to compile this .m file:

```
function hello(name)
  strcat({'Hello '}, name)
end
```

This operation compiles it (this takes some time):

```
DISPLAY="" mcc -m -v -R '-nojvm, -nodisplay' -o hello hello.m
```

If you have some other .m files that you need to include, you need to explicitly specify their location as follows:

```
mcc -m -v -R '-nojvm, -nodisplay' -I /path/to/functions/ -I
/path/to/other/functions/ [...]
```

The resulting files are a text file named readme.txt, a script named run\_hello.sh and an executable named hello.

2025/10/29 04:40 21/33 Applications on the clusters

You can then launch the executable hello like any other executable using a sbatch script:

```
#!/bin/bash

#SBATCH --partition=debug-cpu
#SBATCH --ntasks=1

module load foss/2016a matlab/2016b

srun ./hello Philippe
```

In this case, Philippe is an argument for the function *hello*. Be careful, arguments are always passed to Matlab as strings.



You do NOT need to specify the Matlab license once compiled, and you are not restricted by the number of available licenses.

Please see compile Matlab on gitlab for some examples and scripts related to Matlab compilation.

#### **Matlab PATH**

If you need to add a directory to the Matlab path, for example to use a toolbox installed by you, please proceed as follow.

Add the needed path recursively to matlab path:

```
addpath(genpath('/home/sagon/tests/matlab/dtb/decoding toolbox v3.991'))
```

Save the current matlab path to the default matlab definition path:

```
savepath('/home/sagon/pathdef.m')
```

You can now access any file from the toolbox directly:

```
demo2_simpletoydata
```

# Matlab java.opts

If you are getting erros such as the one listed here when using Matlab through X2go, you can try the following changes to mitigate the issue.

Create a file named java.opts in the default startup folder which is the folder from which you started MATLAB, *i.e.* the folder where you type the matlab command (usually your \${HOME}} base folder).

Put the following content in it:

```
-Dsun.java2d.xrender=false
```

And restart Matlab. This will probably reduce the graphical performance but should not impact the computation time.

#### **CHROMIUM mailbox/texture errors**

If you are getting erros such as the one listed here and here when using Matlab through X2go, unfortunately we have not found a *once-and-for-all* solution yet.

Please contact us providing the MATLAB version as well as the `salloc` full command.

#### **Wavelab**

To use the Wavelab library with Matlab, load Matlab 2014:

```
module load matlab/2014b
```

Launch Matlab as usual and type this command to go to the Wavelab library:

```
cd /opt/wavelab/Wavelab850/
Wavepath (answer /opt/wavelab/Wavelab850)
```

# **OpenCL**

You can use OpenCL on CPU. To compile your software, please proceed as following:

```
gcc -I/opt/intel/opencl-1.2-sdk-6.3.0.1904/include/ -
L/opt/intel/opencl-1.2-sdk-6.3.0.1904/lib64/ -Wl,-
rpath,/opt/intel/opencl/lib64/ -lOpenCL -o hello hello.c
```

# **Distant Paraview**

Thanks to Orestis for this tutorial.

#### Warning:

- Do not use at any point X11 forwarding it will be done by paraview itself.
- You must have the SAME version of paraview on your local machine and on baobab (5.3.0).

Baobab connection:

```
ssh login2.baobab.hpc.unige.ch
```

2025/10/29 04:40 23/33 Applications on the clusters

Get some resources in interactive mode (4 cores here, you can add a lot more options here if you want). You can even ask for GPU nodes (more on this afterwards):

```
salloc -n 4
```

Determine on which node your resources are affected (here node001):

```
echo $SLURM_JOB_NODELIST

SLURM_JOB_NODELIST=node001
```

On another terminal (in your pc) open an ssh tunnel to your NODE (in this case node001):

```
ssh -L 11150:node001:11111 login2.baobab.hpc.unige.ch
```

On the first terminal load paraview (if not loaded by default) and launch pserver:

```
module load foss/2016b ParaView/5.3.0-mpi
srun pvserver --server-port=11111
```

If you asked for GPU nodes the command is slightly different:

```
srun pvserver --server-port=11111 -display :0.0 --use-offscreen-rendering
```

On your local machine launch paraview and click on Connect. There you should find the menu to add a server. Put the name you want in Name, leave Client/Server as Server Type, and `localhost` in Host. The only very important configuration is the port which should be 11150 (the same number as in `11150:node001` from before). Save the configuration (click on Configure and Save, leave startup as Manual) and the click on Connect. The remote paraview session should start immediately. There should be an error message: *Display is not accessible on the server side. Remote rendering will be disabled.* This message is normal.

# **Python**

Default Python version on the cluster is Python 2.7.5.

You can have access to modern Python through Module - Imod

The Python version provided by module come with a lot of packages already installed.

You can check with module spider what are the packages provided. Example:

```
deap-1.3.0, decorator-4.4.0, docopt-0.6.2, docutils-0.15.2, ecdsa-0.13.2,
future-0.17.1, idna-2.8, imagesize-1.1.0, importlib metadata-0.22,
ipaddress-1.0.22, Jinja2-2.10.1, joblib-0.13.2, liac-arff-2.4.0,
MarkupSafe-1.1.1, mock-3.0.5, more-itertools-7.2.0, netaddr-0.7.19,
netifaces-0.10.9, nose-1.3.7, packaging-19.1, paramiko-2.6.0,
pathlib2-2.3.4,
paycheck-1.0.2, pbr-5.4.3, pip-19.2.3, pluggy-0.13.0, psutil-5.6.3,
py-1.8.0,
py expression eval-0.3.9, pyasn1-0.4.7, pycparser-2.19, pycrypto-2.6.1,
Pygments-2.4.2, PyNaCl-1.3.0, pyparsing-2.4.2, pytest-5.1.2, python-
dateutil-2.8.0, pytz-2019.2, requests-2.22.0, scandir-1.10.0,
setuptools-41.2.0, setuptools scm-3.3.3, six-1.12.0, snowballstemmer-1.9.1,
Sphinx-2.2.0, sphinxcontrib-applehelp-1.0.1, sphinxcontrib-devhelp-1.0.1,
sphinxcontrib-htmlhelp-1.0.2, sphinxcontrib-jsmath-1.0.1, sphinxcontrib-
qthelp-1.0.2, sphinxcontrib-serializinghtml-1.1.3, sphinxcontrib-
websupport-1.1.2, tabulate-0.8.3, ujson-1.35, urllib3-1.25.3,
virtualenv-16.7.5, wcwidth-0.1.7, wheel-0.33.6, xlrd-1.2.0, zipp-0.6.0
```

If you need numpy, SciPy, pandas, mpi4py, they are provided by the SciPy-bundle module.

Example to load a recent version of Python with SciPy:

```
ml GCC/8.2.0-2.31.1 OpenMPI/3.1.3 Python/3.7.2 SciPy-bundle/2019.03
```

# **Custom Python lib**

If you need to install a python library or a different version of the ones already installed, virtualenv is the solution.

Python-virtualenv is installed on Baobab http://www.virtualenv.org/en/latest/

Begin by loading a version of python using module (see above)

Create a new virtualenv if it's not already existing (put it where you want and name it like you want):

For virtualenv 20 or above (starting from Python/3.8.2):

```
capello@login2:~$ virtualenv ~/baobab_python_env
```

For previous virtualenv versions:

```
capello@login2:~$ virtualenv --no-site-packages ~/baobab_python_env
```

This will create a directory named baobab\_python\_env in your home directory.

Every time you want to use your virtualeny, you should activate it first:

```
. ~/baobab_python_env/bin/activate
```

Install all the needed packages in the environment:

2025/10/29 04:40 25/33 Applications on the clusters

```
~/baobab python env/bin/pip install mpi4py
```

Use your new environment:

```
~/baobab_python_env/bin/python
```

# Pip install from source

By default when you use pip to install a library, it will download a binary of .whl file instead of building the module from source. This may be an issue if the module itself depend on a custom libc version or is optimized for a given kind of CPU. In this case, you can force pip to install a module by building it from source.

Example to build hpy from source. The h5py argument to -no-binary is to specify that you want to build from source only h5py.

```
pip install --no-binary h5py h5py
```

# R project and RStudio

The latest version of R (if it's not the latest, you can ask us to install it) is installed on the cluster.

Please see here for an sbatch example with R. You will find as well an exemple using the R package parallel.

External helper to create sbatch scripts for R golembash

#### **RStudio**

RStudio is IDE (Integrated Development Environment) for R, basically a more user friendly version than plain R.

With the Baobab upgrade to CentOS 7 (cf.

https://hpc-community.unige.ch/t/baobab-migration-from-centos6-to-centos7/361 ) we do not provide anymore a central RStudio.

Instead, you can download the upstream Open Source binary RStudio Desktop version (cf. https://rstudio.com/products/rstudio/download/) and directly use it, here the instructions:

1. install it in your \${HOME} folder:

```
capello@login2:~$ mkdir Downloads
capello@login2:~$ cd Downloads
capello@login2:~/Downloads$ wget ${URL_FOR_rstudio-${VERSION}-x86_64-
fedora.tar.gz}
[...]
capello@login2:~/Downloads$ tar axvf rstudio-${VERSION}-x86_64-
```

```
fedora.tar.gz
[...]
capello@login2:~/Downloads$
```

- 2. launch an interactive graphical job:
  - 1. connect to the cluster using GUI access / Desktop with X2Go or using ssh -Y from a machine with an X server such as Linux or Mac.
  - 2. start an interactive session on a node (see Interactive Slurm jobs):

```
capello@login2:~$ salloc -p debug-cpu -n 1 -c 16 --x11
salloc: Pending job allocation 39085914
salloc: job 39085914 queued and waiting for resources
salloc: job 39085914 has been allocated resources
salloc: Granted job allocation 39085914
capello@node001:~$
```

Doing so, you will have 16 cores on one node of the partition debug-cpu for a max time of 15 minutes. Specify the appropriate duration time, partition, etc. like you would do for a normal job.

3. load one of the R version supported by RStudio, for example:

```
capello@node001:~$ module spider R/3.6.0

R: R/3.6.0

Description:
R is a free software environment for statistical computing and graphics.

You will need to load all module(s) on any one of the lines below
before the "R/3.6.0" module is available to load.

GCC/8.2.0-2.31.1 OpenMPI/3.1.3

[...]

capello@node001:~$ module load GCC/8.2.0-2.31.1 OpenMPI/3.1.3

capello@node001:~$ module load PostgreSQL/11.3-Python-3.7.2

capello@node001:~$ module load R/3.6.0

capello@node001:~$
```

4. run RStudio:

```
capello@node001:~$ ~/Downloads/rstudio-${VERSION}/bin/rstudio
```

2025/10/29 04:40 27/33 Applications on the clusters



Latest version of Rstudio needs an aditional dependency loaded

module load PostgreSQL/11.3-Python-3.7.2

## R packages

You can install R packages as a user. Just follow once the steps given below:

Create a file named .Rprofile (note the dot in front of the file) in your home directory with the following content:

```
local({
r = getOption("repos") # hard code the Switzerland repo for CRAN
r["CRAN"] = "https://stat.ethz.ch/CRAN/"
options(repos = r)
})
```

Create a file named .Renviron (note the dot in front of the file) in your home directory with the following content:

```
(yggdrasil)-[alberta@login1 ~]$ cat ~/.Renviron
R_LIBS=~/Rpackages/
```

Create a directory where to store the installed R packages:

```
(yggdrasil)-[alberta@login1 ~]$ mkdir ~/Rpackages
```

Once done, make sure you have loaded R with module. Then, from a R command line you can install a R package.

For R version 3.5 and below:

```
install.packages("ggplot2")
```

For R version 3.6 or above:

```
Sys.setenv(R_INSTALL_STAGED = FALSE)
install.packages("ggplot2")
```

Use your newly installed package:

```
library(ggplot2)
```

#### **Example:**

```
(yggdrasil)-[alberta@login1 scratch]$ cat sbatch.sh
#!/bin/sh
```

```
#SBATCH --partition=shared-cpu
#SBATCH --time=0:05:00
module load GCC/11.3.0 OpenMPI/4.1.4 R/4.2.1
srun R CMD BATCH test.r
```

```
(yggdrasil)-[alberta@login1 scratch]$ cat test.r
##### Set the working directory #####
#setwd("/srv/beegfs/scratch/users/a/alberta")

##### Clear the global environment ####
rm(list = ls())

Sys.setenv(R_INSTALL_STAGED = FALSE)

##### Install and load packages ####
install.packages("rootSolve")
library(rootSolve)
```

```
(yggdrasil)-[alberta@login1 scratch]$ sbatch sbatch.sh
sbatch sbatch.sh
Submitted batch job 25456882
(yggdrasil)-[alberta@login1 scratch]$ sac -j 25456882
                   JobName
          JobID
                              Account
                                           User
                                                       NodeList
                                                                   NTasks
                      End
Start
                               State
       25456882 sbatch.sh
                                burgi
                                        alberta
                                                         cpu149
2023-07-14T10:25:13 2023-07-14T10:25:29 COMPLETED
```

The log file is test.r.Rout and you should see everything is working:)

# **Variant Effect Predictor (VEP)**

This tutorials is inspired from VEP documentation: https://www.ensembl.org/info/docs/tools/vep/script/vep\_download.html#singularity

According vep maintainers (https://github.com/Ensembl/ensembl-vep/issues/1515) each users should have one instance of vep. Please edit destination use the following sbatch to install your instance.

```
To install species edit the following option in the install cmd line:

"-a cfp -s <cpecies_name> "

(baobab)-[alberta@login2 ~]$ mkidr $HOME/vep_singularity
(baobab)-[alberta@login2 vep_singularity]$ cd $HOME/vep_singularity
(baobab)-[alberta@login2 vep_singularity]$ vim install_vep.sh

#!/bin/bash
```

2025/10/29 04:40 29/33 Applications on the clusters

```
#SBATCH --job-name=install_vep
#SBATCH --output=output.log
#SBATCH --partition=shared-cpu
#SBATCH --time=01:00:00
#SBATCH --cpus-per-task 12
#SBATCH --mem=12GB
#SBATCH --chdir=/scratch
#SBATCH --export=All

srun singularity pull --name vep.sif docker://ensemblorg/ensembl-vep
srun mkdir vep_data
srun singularity exec vep.sif INSTALL.pl -c vep_data -a p -g all -r vep_data
srun mkdir -p $HOME/vep_singularity/install_dir
srun mv * $HOME/vep_singularity/install_dir
```

## **Install species**

To download species after the installation run the following cmd line(it assumes you have the same directory arch as above):

It ran the install command on debug-cpu slurm partition assuming the install required less than 15 min otherwise use another partition with -time=HH:mm:ss

```
(baobab)-[alberta@login2 ~]$ cd
/home/users/a/alberta/vep_singularity/install_dir
(baobab)-[alberta@login2 ~]$ srun singularity exec vep.sif INSTALL.pl -c
vep_data -a cf -s <species_name>
```

# **Apptainer (was Singularity)**

#### Intro

Apptainer is a Docker like for HPC. It is available directly on the OS, not through module. As you don't have root access on Baobab, you cannot build recipe file on Baobab. If you need to do this, you may want to build the image on your own machine and transfer the image to Baobab. You can download existing images from shub (singularity hub) or from docker (docker public or private registry).

The image will be converted to a read-only squashfs on Baobab disk. It's not possible to have writable images as user. Instead you should build the image as sandbox.

Apptainer images are immutable, but it is possible to append an overlay, see below.

#### Pull an existing image





Do not run the following commands on login node!! It's using too much cpu

Exemple with Rstudio:

#### Create a project directory and pull the image from compute node:

```
(baobab)-[alberta@login2 ~]$ MYPROJECT="rocker"
(baobab)-[alberta@login2 ~]$ mkdir -p singularity/$MYPROJECT
(baobab)-[alberta@login2 ~]$ cd !$
(baobab)-[alberta@login2 rocker]$ salloc --partition=shared-cpu --
time=00:30:00 --cpus-per-task 12
(baobab)-[alberta@cpu300 rocker]$ apptainer pull docker://rocker/rstudio:4.2
(baobab)-[alberta@cpu300 rocker]$ ls
rstudio_4.2.sif
```

And Voila; I get my sif image rstudio 4.2.sif

# Convert a Docker image

Example to download an existing docker image and build a squashfs image from it:

```
apptainer build lolcow.simg docker://godlovedc/lolcow
```

Then you will end-up with an Apptainer image named lolcow.simg.

Example to download an image from a custom registry:

apptainer build docker://registry.gitlab.com/flowkit/webservice/compute



By default, Apptainer will build your image in `/tmp`. This is an issue on the login node because `tmp` is small. You should instead specify an alternate `tmp` https://apptainer.org/docs/user/main/build\_env.html#temporary-folders

#### Run a container

Once you have a singularity image, you can do various things:

run a container (the script named singularity at the root of the image, proceed as follow

```
apptainer run lolcow.simg
```

When you import a docker image, the ENTRYPOINT is used to create the singularity run script.

• exec to execute a single command inside the container.

2025/10/29 04:40 31/33 Applications on the clusters

• shell to launch a container and to spawn a shell (/bin/bash)

# **Modify the image (not persistent)**

If you need for example to install a new rpm inside the image, you can use an ephemeral overlay. In the example below, we are using —fakeroot to behave as if we were root inside the container.

```
apptainer exec --fakeroot --writable-tmpfs [...]
```

Here you can install an rpm for example, but as soon as you close the container, it is "reset".



if you are on a compute node, the variable \$TMPDIR is set to /scratch. You should either unset it or mount /scratch

# **Modify the image (persistent)**

The way to go with apptainer is to add a writable overlay. This is an ext3 fs. You can create for example a sparse 10G image like that:

```
(baobab)-[sagon@cpu065 ~]$ apptainer overlay create --fakeroot --sparse --size 10000 my_overlay.img
```

Notice we added the —fakeroot flag. If you don't, you won't be able to use the image when apptainer is started with —fakeroot

You can then start apptainer

```
(baobab)-[sagon@cpu065 ~]$ apptainer exec --fakeroot --overlay
my_overlay.img [...]
```

#### References

For more examples, please check the following posts :

- [tutorial] launch openpose with GPU support through Singularity
- Help needed running MPI/Palabos software using Singularity
- · Quick guide to Docker, singularity and shifter

# Stata

Stata versions

\* 14 mp 24 cores \* 16 mp 32 cores \* 17 mp 32 cores

are availables on the cluster.

To use it, you need to add load Stata using module. Example to load Stata 17:

```
module load Stata/17
```

The Stata binaries are stata-mp or xstata-mp for the graphical interface.

If you need a graphical interactive session, please proceed as follows from x2go for example:

```
salloc -n1 -c 16 --partition=interactive-cpu --time=15:00 --x11 srun -n1 -N1
--pty $SHELL
xstata-mp
```

Doing so will launch a graphical Stata on a debug node with 16 cores for a 15 minutes session. See on this document for other partition/time limits.



Please keep in mind that the cluster may be full and that you will have to wait until the resources are allocated to you. It's best to launch stata in batch mode instead of using it interactively.

To launch Stata in batch mode, see the Multithreaded jobs section and specify that you want one task and n cpus.

Please see here for an sbatch example with Stata.

# **TensorFlow**

Please see TensorFlow on gitlab for some examples and scripts related to TensorFlow.

**ATTENTION**, the module:TensorFlow we provide are compiled with GPU support (see https://www.tensorflow.org/install/source#gpu\_support), thus they must be used on a GPU partition (cf. Gpu resources and Gpu jobs section).

You can also read this post :

Getting started with TensorFlow on Baobab (ImportError: libcuda.so.1)

# Compile and install a software in your /home

The following posts can inspire you if you need to compile or install a software in your \$HOME directory :

- [HOWTO] compile a software in your home directory
- [howto] install and run cudimot

2025/10/29 04:40 33/33 Applications on the clusters

• Remember, you **do not** have sudo rights. Rules and etiquette

From:

https://doc.eresearch.unige.ch/ - eResearch Doc

Permanent link:

https://doc.eresearch.unige.ch/hpc/applications\_and\_libraries?rev=1744385512



