JADE CLUSTER MANUAL

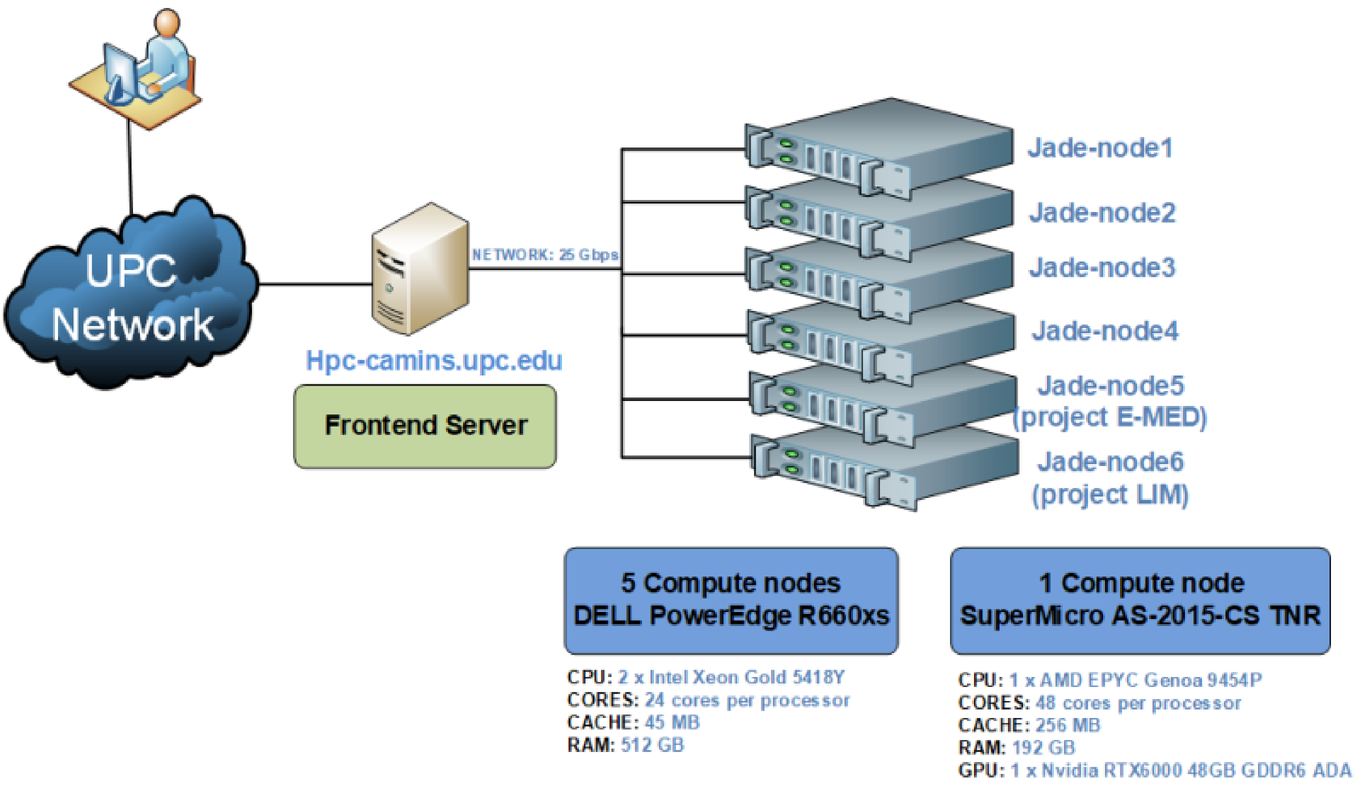
This document is intended to give some basic notes in order to work with the JADE High Performance Green Computing Cluster of the Civil Engineering School (ETSECCPB) located in the CaminsTECH facilities.

1.- ARCHITECTURE

JADE is a **high performance green computer cluster** formed by a set of Dell and SuperMicro servers. It is a multitasking and multiuser environment configured with **Ubuntu** Linux operating system and uses the **SLURM** open-source workload manager (slurm.schedmd.com). JADE is oriented towards users who need to run programs with high computing performance requirements of memory and processing time. The cluster has two main parts:

- A **frontend/access server**:
  - This is the server where users can launch their simulations (jobs), but they actually run on the computing servers. Every user has his personal workspace and useful tools such as compilers, editors, etc.
  - Name: **hpc-camins**
- **Six compute servers/nodes (5 nodes of 48 Intel cores [jade1…jade5] and 1 node of 48 AMD cores and GPU Nvidia RTX 6000 [jade6])**:
  - Every compute server executes submitted jobs from the fronted server.
  - Name: **Jade-nodeN**, where N is the number of the compute server (e.g. 1, 2…)

As a **Green Computing Cluster**, the compute servers are powered on while jobs are running (when a node is idle for 30 minutes it will be powered off). An overview of the architecture of JADE and its characteristics is shown below:


UPC Network
NETWORK: 25 Gbps
Hpc-camins.upc.edu
Frontend Server
Jade-node1
Jade-node2
Jade-node3
Jade-node4
Jade-node5 (project E-MED)
Jade-node6 (project LIM)
5 Compute nodes DELL PowerEdge R660xs
CPU: 2 x Intel Xeon Gold 5418Y
CORES: 24 cores per processor
CACHE: 45 MB
RAM: 512 GB
1 Compute node SuperMicro AS-2015-CS TNR
CPU: 1 x AMD EPYC Genoa 9454P
CORES: 48 cores per processor
CACHE: 256 MB
RAM: 192 GB
GPU: 1 x Nvidia RTX6000 48GB GDDR6 ADA

2.- ACCES MODES

In order to work with JADE cluster, it is required that the user asks for an account with the UTGAC IT Services (https://caminstech.upc.edu/es/serveis/calculintensiu). Once the account is created, UPC users will be able to gain access to the service with their UPC username and password, otherwise a local username and password will be given. Connection to the server is only allowed from computers in Camins networks (buildings B0, B1, B2, C1, C2, D1, D2 and the computer rooms of the Civil Engineering School) or from anywhere using the UPCLink VPN client (VPN UPC Manuals).

2.1.- Terminal Access

You can use any tool that supports **Secure Shell (SSH)** to connect to the service.

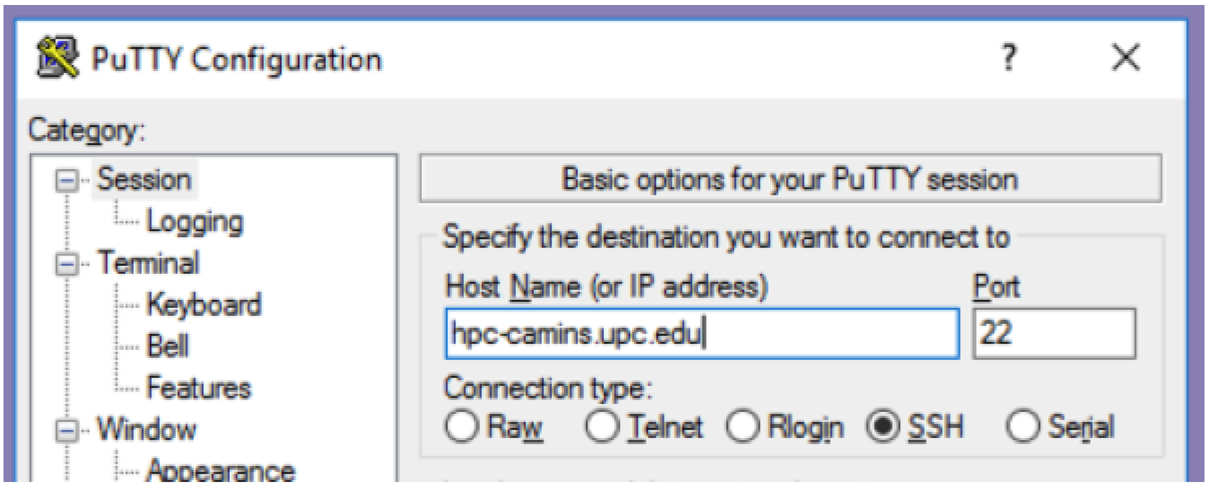
- In **Linux** almost all distributions have a preinstalled ssh client.
- In **MacOSX** there is a ssh client in Terminal.
- In **Windows** for example, use Putty (ssh client) or Mobaxterm (cygwin terminal with a SSH and SFTP client that includes Graphical Access).

Putty

Configure connection as follows:

Host name	hpc-camins.upc.edu
Port number	22

User name	UPC_username
Password	UPC_password



Host Name (or IP address): hpc-camins.upc.edu
Port: 22
Connection type: [✓ SSH]

2.2.- Graphical Access

To run programs using graphical output, such as Abaqus CAE or Matlab, use:

- Mobaxterm (cygwin terminal with a SSH and SFTP client)
  - http://mobaxterm.mobatek.net/
- x2go client
  - http://wiki.x2go.org/doku.php/doc:installation:x2goclient

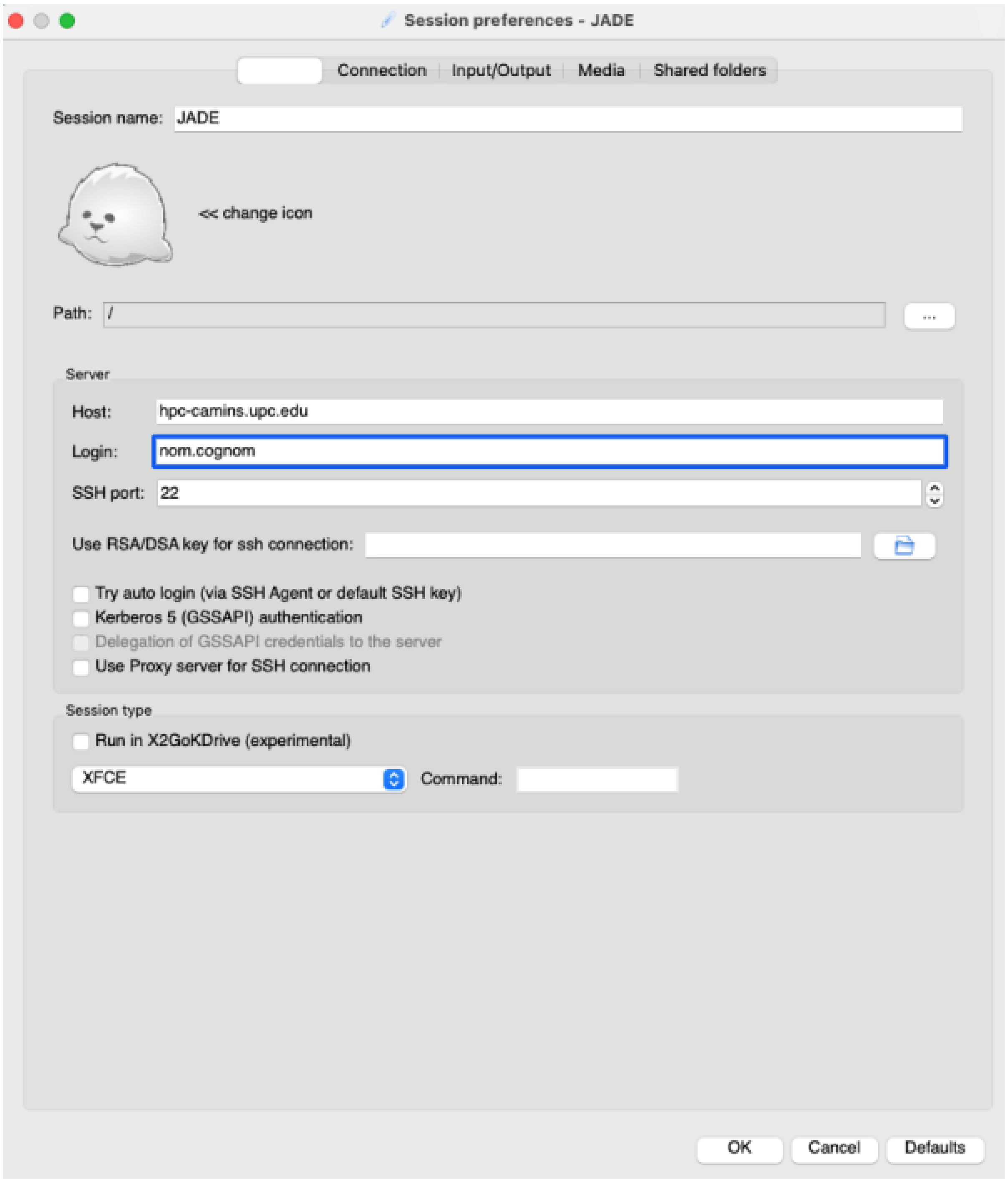
Both options (terminal and graphical access) use the SSH protocol for secure connections.

X2go:

Add a New Session with the parameters in the table below.

Host / IP	hpc-camins.upc.edu / 147.83.87.20
Port (TCP)	22
Services	SSH, SFTP and x2go
Authentication method	username/password
Desktop	XFCE

Your preferences screen should look like this:



Session name: JADE

Path: /

Server

Host: hpc-camins.upc.edu

Login: nom.cognom

SSH port: 22

Use RSA/DSA key for ssh connection:

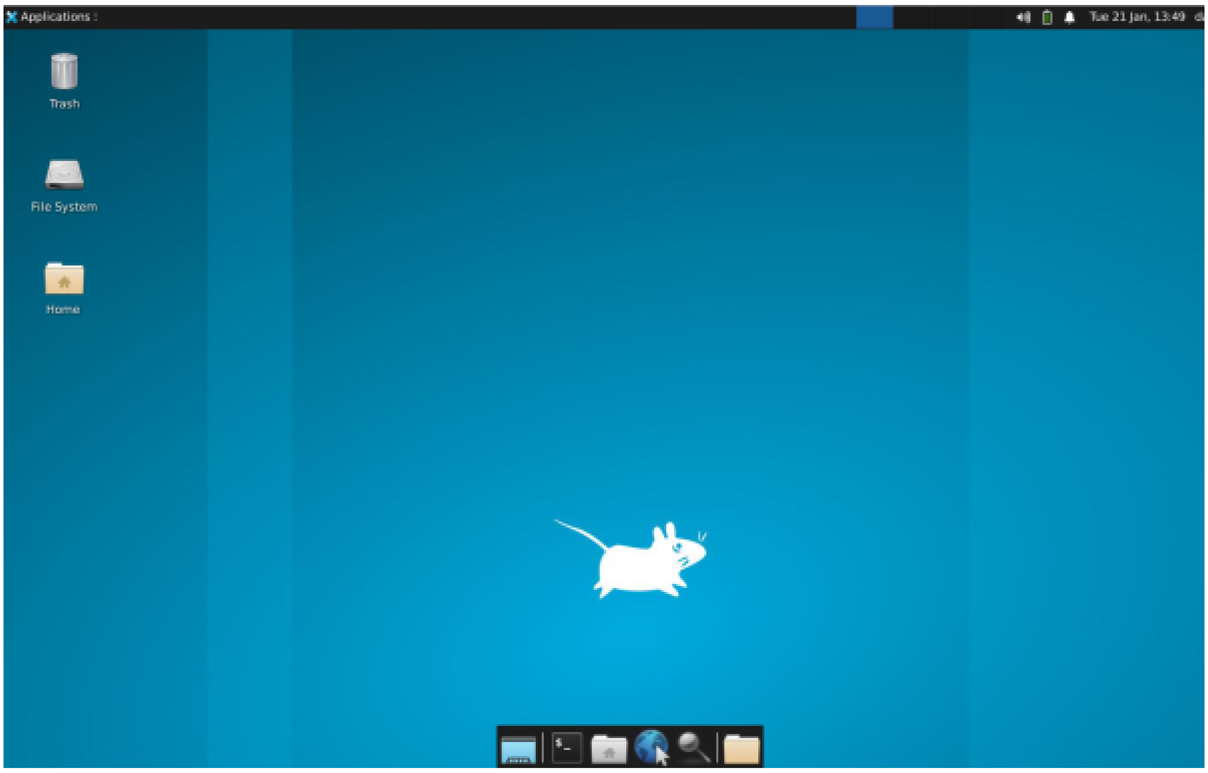
- [ ] Try auto login (via SSH Agent or default SSH key)
- [ ] Kerberos 5 (GSSAPI) authentication
- [ ] Delegation of GSSAPI credentials to the server
- [ ] Use Proxy server for SSH connection

Session type

- [ ] Run in X2GoKDrive (experimental)

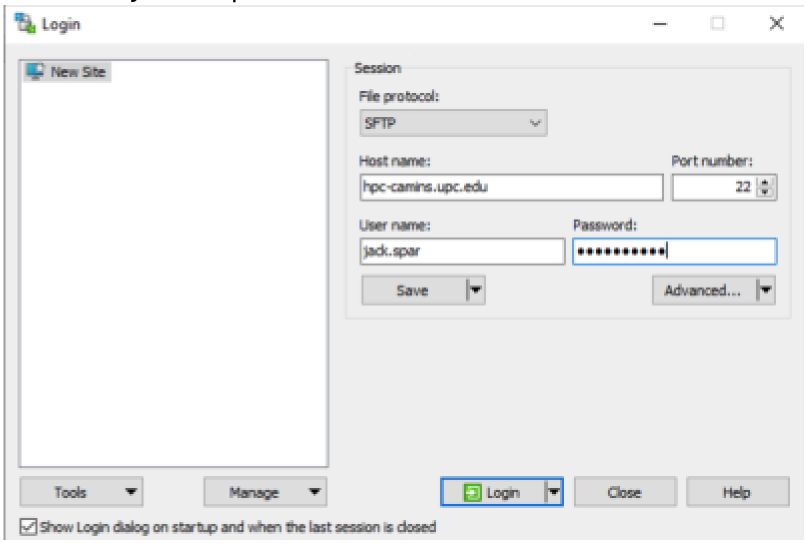
XFCE Command:

Once you provide your username and password you will see your personal desktop:



2.3.- File Transfer

In order to download/transfer files from/to JADE (hpc-camins), it is recommended to use a tool such as winSCP configuring it as follows.



Session
File protocol: SFTP
Host name: hpc-camins.upc.edu
Port number: 22
User name: jack.spar
Password: •••••••••••
Show Login dialog on startup and when the last session is closed

3.- LINUX BASIC COMMANDS

JADE Cluster is a Ubuntu 22.04 Linux operating system. These are some basic commands in a Linux environment (shell) that you will find useful when working on it.

Command	Description	Example
pwd	shows the current working directory	`$ pwd`
mkdir	creates a new directory	`$ mkdir directory`
cd	changes from one directory to the one specified	`$ cd directory`

Command	Description	Example
ls	lists the files and directories under the current directory	`$ ls`
mv	moves files or subdirectories from one directory to any other specified or renames the file or directory	`$ mv test.txt /home/directory/`
cp	copies files into the specified directory	`$ cp test.txt /home/user/`
rm	deletes a file NOTE: once deleted it cannot be recovered	`$ rm test.txt`
rmdir	deletes the specified directory which must be empty	`$ rmdir directory`
more, cat	allows file visualization, not modifications allowed	`$ more test.txt`, `$ cat test.txt`

For more detailed information about Linux Shell commands take a look at http://linuxcommand.org/lc3_learning_the_shell.php

4.- DISK SPACE

Each user has a **personal disk space** to store files in JADE. The user's home directory has the following path:

```
/users/USERNAME
```

It is recommended to use this space to store code, executable and data files. The initial disk quota assigned is 25 GB per user; however should you need to increase the amount of space, it is possible to request this from UTGAC IT Services (https://caminstech.upc.edu/es/on-trobar-nos). To check the available quota use the next command:

```
 > quota -s
```

Additionally, every user has **temporary disk space (scratch)** available.

4.1.- NFS scratch personal directory

```
 /scratch/USERNAME
```

This directory does not have any quota restrictions, for this reason, it is suggested to use this temporary scratch directory to run jobs sent to the batch queues and to store execution results. All compute nodes have access to this NFS mounted scratch directory. The amount of space available is 5 TB and it is shared by all users in the cluster, therefore it is requested that users consider others by not overloading the disk. In the case that a user occupies too much space and affects the jobs of others, **the system administrator could delete files without notice. Please note: files in the scratch not changed in more than 32 days will be deleted automatically every day at 11 PM.**

5.- SOFTWARE

5.1.- Compilers

See Changing software environment section to choose the compiler version in use.

Fortran compilers installed:

COMPILER	COMMAND
Intel Fortran Compiler (OneAPI 2024.2.1 -default-)	ifx
GNU Fortran compiler (11.0.4 -default-)	f95, gfortran

C/C++ compilers installed:

COMPILER	COMMAND
Intel C++/DPC++ compiler (OneAPI 2024.2.1 -default-)	icx
Intel Data Parallel C++ compiler (OneAPI 2024.2.1)	icpx
GNU project C and C++ compiler (11.4.0 -default-)	cc, gcc, g++

Additionally installed, is the Intel HPC Toolkit (version 2024.2) which include the following tools:

- Intel MPI Library
- Intel Math Kernel Library
- Intel Debugger for Linux
- Intel VTune

NOTE: Remember to load your selected compiler software environment before to use it. (Section 5.3)

5.2.- Parallel Computing

The Intel and GNU the MPI libraries (Message Passing Interface) are installed in TITANI through its OpenMPI implementation.

See Changing software environment section to choose the compiler version in use.

These libraries are interfaces for parallel computing designed for programs that take advantage of multiple cores.

Compilation Commands using OpenMP (only shared memory):

Use the following commands to compile a program prepared for parallelization with OpenMP:

GNU Compilers:

Remember to use modules tool to define the GNU Compiler Version to use. The module should be loaded before compilation and before execution within your batch script.

- **GNU Fortran:**

```
f95 –o program_name –fopenmp program_code.f

gfortran –o program_name –fopenmp program_code.f
```

- **GNU C/C++:**

```
gcc –o program_name –fopenmp program_code.c

g++ –o program_name –fopenmp program_code.cpp
```

Intel Compilers (Intel OneAPI 2024.2.1 version):

- **Intel Fortran:**

```
 ifx –o program_name –qopenmp program_code.f90
```

- **Intel C/C++/DPC++:**

```
icx -o program_name –qopenmp program_code.c

icpx –o program_name –qopenmp program_code.cpp
```

Compilation Commands using MPI (shared and distributed memory):

Use the following commands to compile a program prepared for parallelization with MPI:

GNU Compilers:

Remember to use modules tool to define the OpenMPI Version to use. The module should be loaded before compilation and before execution within your batch script.

- **GNU Fortran MPI:**

```
mpif90 –o program_name program_code.f
```

- **GNU C/C++ MPI:**

```
mpicc –o program_name program_code.c

mpicxx –o program_name program_code.cpp
```

Intel Compilers (Intel OneAPI 2024.2.1 version):

- **Intel Fortran:**

```
 mpiifx –o program_name program_code.f
```

- **Intel C/C++:**

```
mpiicx –o program_name program_code.c

mpiicpx –o program_name program_code.cpp
```

Execute the following command to display more help regarding this commands:

```
 $ command -help
```

5.3.- Changing Software Environment

5.3.1.- Using Modules

As seen, different applications and compilers are installed into the JADE cluster. Initially, when a user logs in the system, Intel Compilers and MPI environment configurations are loaded and ready to be used by default.

In the future if new software versions that require new environment variables are installed it may be necessary to load new configuration profiles.

You can list the available configurations with the command «module avail». It returns a huge list of modules that are installed in the system. This list contains the most used modules:

```
> module avail
------- /users/software/easybuild/modules/all -------
   Boost/1.81.0-GCC-12.2.0
   Boost/1.83.0-GCC-13.2.0                     (D)

   CMake/3.24.3-GCCcore-12.2.0
   CMake/3.26.3-GCCcore-13.1.0
   CMake/3.27.6-GCCcore-13.2.0                 (D)
   CUDA/11.7.0
   CUDA/12.0.0
   CUDA/12.3.0                                 (D)
                (D)
   FFTW.MPI/3.3.10-gompi-2022b
   FFTW/3.3.10-GCC-12.2.0
   FFTW/3.3.10-GCC-13.2.0                      (D)

   FlexiBLAS/3.2.1-GCC-12.2.0
   FlexiBLAS/3.3.1-GCC-13.2.0                  (D)

   GCC/11.1.0
   GCC/12.2.0
   GCC/13.1.0
   GCC/13.2.0
   GCC/13.3.0                                  (D)

   HDF/4.2.15-GCCcore-12.2.0
   HDF5/1.14.0-gompi-2022b
   Java/11.0.20
   jq/1.6-GCCcore-12.2.0
   json-c/0.16-GCCcore-12.2.0
   libGLU/9.0.2-GCCcore-12.2.0

   OpenBLAS/0.3.21-GCC-12.2.0
   OpenBLAS/0.3.24-GCC-13.2.0                  (D)

   OpenMPI/4.1.4-GCC-12.2.0
   OpenMPI/5.0.3-GCC-13.3.0                    (D)

   OpenSees/3.7.0
   Perl/5.36.0-GCCcore-12.2.0
   Perl/5.38.0-GCCcore-13.2.0
   Perl/5.38.0
   Perl/5.38.2-GCCcore-13.3.0                  (D)

   Python-bundle-PyPI/2023.10-GCCcore-13.2.0
   Python/3.10.8-GCCcore-12.2.0-bare
   Python/3.10.8-GCCcore-12.2.0
   Python/3.11.5-GCCcore-13.2.0
   Python/3.12.3-GCCcore-13.3.0                (D)

   ScaLAPACK/2.2.0-gompi-2022b-fb
   SciPy-bundle/2023.02-gfbf-2022b
   SciPy-bundle/2023.11-gfbf-2023b             (D)

   virtualenv/20.24.6-GCCcore-13.2.0
```

You can load one of them:

```
> module load OpenMPI/5.0.3-GCC-13.3.0
```

You can back to the initial environment:

```
> module purge
```

5.3.2.- Intel OneAPI

The Intel OneAPI environment is loaded by default. No command is needed.

5.4.- Running Jobs: batch queues (partitions)

JADE uses the SLURM open-source workload manager to manage cluster resources (nodes, processors, cores, memory, GPU). It provides a queue management system oriented to send jobs for their execution and uses a job scheduler to allocate the simulations to the compute nodes. The job scheduler sends the job from the HPC-CAMINS (Jade frontend) server to any compute node depending on the instant cluster resource usage. When all resources are busy, newer jobs will wait until older jobs finish their processes. Therefore, to execute a job it is required to send it to a predefined batch queue according the program specific needs. The batch queues (called partitions in SLURM) are described below in detail as well as its maximum CPU run time, CPU maximum wall time, maximum number of cores allowed, maximum number of simultaneous jobs allowed to be executed or queued, memory limitations and other restrictions. Consider the following recommendations to send jobs to the queues:

- Each queue imposes restrictions per user for the limit of executing and pending jobs.
- It is forbidden to link jobs execution that means the same shell script executes one job after another executed before.
- Contact the UTG IT Services to ask for access to restricted queues in the particular case you need to increase the limits imposed for each queue.

Queues' characteristics could be modified regarding its maximum number of jobs and the memory limits according the users' needs, the computers' workload, among other circumstances, therefore it is recommended to consult the following link to have the most up to date characteristics:

PENDENT———»»»»»»»> http://caminstech.upc.edu/faq/titani

There are four queues (partitions) defined in JADE: two common queues: serial and parallel, and two dedicated queues: lim-gpu and emed.

- SERIAL queue: This is the default queue and it only allows to submit serial jobs, namely that only use a core in their execution. Every user can run 15 jobs simultaneously and has 10 pending jobs.
- PARALLEL queue: This queue only allows to submit parallel jobs that is to say that use more than a core in their execution. Every user can run 3 jobs simultaneously and has 3 pending jobs. It is allowed to use 96 cores per user, in a job or in total with all his jobs.
- LIM-GPU queue: This queue is only for members of the lim-gpu group, and it has no limit on time or the number of jobs. The only resources available are 48 cores and 1 GPU in jade-node6.
- EMED queue: This queue is only for members of the emed group, and it has no limit on time or the number of jobs. The only resources available are 48 cores in jade-node5.

The Serial queue uses jade-node1 to jade-node5 computing nodes (240 cores) and a maximum of 48 cores to execute their jobs. The Parallel queue uses jade-node1 to jade-node5 computing nodes (240 cores) and a maximum of 192 cores can be used.

The following table reflects a summary of the characteristics of each queue. This information has been retrieved by the time this tutorial has been created, consult the most up to date information at the link mentioned before.

QUEUE	USER RESTRICTIONS		JOB LIMITS	
	Max Number of running (pending) jobs	Max Number of cores	Max CPU Runtime Minutes, (Hours), [Days]	Memory limit
serial	15 (10)	-	20160, (336), [14]	64 GB/job
parallel	3 (3)	96	138240, (2304), [96]	768 GB/job, 384 GB/node
			Wall Time 10080, (168), [7]	

5.4.1.- Sending jobs

As mentioned before, to run simulations at JADE is necessary to send jobs to the computing queues. In fact, it is not allowed to execute high CPU usage on the frontend server because it has not enough resources to do theses taks. The **sbatch** command is used to send jobs to a defined queue of JADE. It is required to make a *shell script*. This script would look like the following:

```
#!/bin/bash
<path>/program_to_execute
```

Where <**path**> is the full name of the directory where the program to run is. Use the following syntax:

```
sbatch –p queue_name <shell_script_name>
```

In the directory where the **sbatch** command was launched, SLURM creates one output file called `slurm-<JOBID>.out` containing all output and errors of the execution. <JOBID> is the own ID of each sent job or job identifier.

By **default**, the **sbatch** command sends a job to the serial queue and asks to the job scheduler for 4096MB of RAM memory. If you need more memory resources you can use this flag:

```
  --mem <amount_of_RAM(MB)>
```

Please, do not ask for more memory than needed.

Example: submit a job myscript.sh asking for a machine 10GB of RAM.

```
 sbatch -p serial –-mem 10000 myscript.sh
```

It is also possible and very useful to specify the job parameters inside the shell script. These are the most used:

```
#!/bin/bash

#SBATCH --partition=<name_of_the_queue>
#SBATCH --job-name=<name_for_the_job>
#SBATCH --output=<output_filename.out>
#SBATCH --error=<error_filename.err>
#SBATCH --mem=<amount_of_RAM(MB)>
#SBATCH --cpus-per-task=<num_of_threads> # OMP programs
#SBATCH --nodes=<num_of_nodes> # MPI programs #SBATCH --ntasks-per-
node=<num_of_cores> # MPI programs
```

5.4.2.- Sending parallel jobs

In order to send jobs for parallel execution, namely using more than one core, it must be used the

parallel queue.

The command used would be the following syntax:

```
 sbatch –p parallel <parallel_script_name>
```

By **default**, the **sbatch** command asks to the job scheduler for 4096MB of RAM memory per node. If you need more memory resources you can use this flag:

```
--mem <amount_of_RAM_per_node(MB)>
```

Using OpenMP (only shared memory):

```
sbatch –p parallel –c <num_threads> omp_script
sbatch –p parallel –-cpus-per-task=<num_threads> omp_script
```

A *omp_script* file would be for instance:

```
#!/bin/bash
<path>/omp_program_to_execute
```

Example: OMP program using 4 threads and 12 GB of RAM in the parallel queue:

```
 sbatch –p parallel –c 4 –-mem=12000 omp_script
```

Example: OMP program using 4 threads to the parallel queue specifying the parameters in the shell script *omp_script*:

```
#!/bin/bash
#SBATCH --partition=parallel
#SBATCH --mem=8000
#SBATCH --cpus-per-task=4
<path>/omp_program_to_execute
```

Shell command:

```
sbatch omp_script
```

Using MPI (shared and distributed memory):

```
sbatch –p parallel –-ntasks-per-node=<num_of_cores> mpi_script
sbatch –p parallel -–nodes=<num_of_nodes>
                     –-ntasks-per-node=<num_of_cores> mpi_script
```

A *mpi_script* file would be for instance:

```
#!/bin/bash
```

```
mpirun <path>/mpi_program_to_execute
```

Example: MPI program using 8 cores in the parallel queue:

```
 sbatch –p parallel –-ntasks-per-node=8 mpi_script
```

Example: MPI program using 8 cores in the parallel queue specifying the parameters in the shell script *mpi_script*:

```
#!/bin/bash
#SBATCH --partition=parallel
#SBATCH –-ntasks-per-node=8
mpirun <path>/mpi_program_to_execute
```

Shell command:

```
 sbatch mpi_script
```

Example: MPI program using 2 nodes and 4 cores per node in the parallel queue:

```
 sbatch –p parallel –-nodes=2 –-ntasks-per-node=4 mpi_script
```

Example: MPI program using 2 nodes: 4 cores and 4 GB per node in the parallel queue specifying the parameters in the shell script:

```
#!/bin/bash
#SBATCH --partition=parallel
#SBATCH --nodes=2
#SBATCH --mem=4000
#SBATCH –-ntasks-per-node=4

mpirun <path>/mpi_program_to_execute
```

```
 sbatch mpi_script
```

5.4.3.- Consulting service and jobs in the queues

The **status of the service** is shown by main commands **sinfo** and **squeue**:

```
 sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
parallel     up   infinite      3  idle~ jade-node[3-5]
parallel     up   infinite      1  alloc jade-node2
parallel     up   infinite      1    mix jade-node1
serial*      up   infinite      4  idle~ jade-node[2-5]
serial*      up   infinite      1    mix jade-node1
emed         up   infinite      1  idle~ jade-node5
```

```
limgpu       up   infinite      1    mix jade-node6
```

This command shows the state of the compute nodes, in this example:

Node jade-node2 is powered on and has been fully allocated. Node jade-node1 is powered on and has been allocated by one or more jobs (but not fulfilled). Nodes jade-node3 to jade-node5 are idle and in power saving mode. Note that "~" means that is powered off (energy saving mode) and "#" that is powering on. To show the current job list use the command **squeue**:

```
 squeue
JOBID PARTITION     NAME       USER       ST   TIME     NODES
NODELIST(REASON)
621   parallel    bucleINT    jack.sparr  R    18:28:08      1    jade-node1
622   parallel    IntelMPI    david.garc  R    18:25:48      1    jade-node1
625   serial      bucleCua    jack.sparr  CF   18:20:26      1     jade-
node2
```

State "R" means that the job is running. Sometimes powered off nodes must to be started in order to process a job, the average up time is about 5 minutes so the state of the job is "CF" (configuring). Please be patient.

To show complete information of your jobs (running or not) use the following command:

```
 scontrol show job 219
JobId=219 JobName=prueba11
   UserId=jack.sparr(20099) GroupId=gr-msr(2023) MCS_label=N/A
   Priority=1 Nice=0 Account=users QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=01:08:46 TimeLimit=7-00:00:00 TimeMin=N/A
   SubmitTime=2025-02-17T12:59:20 EligibleTime=2025-02-17T12:59:20
   AccrueTime=2025-02-17T12:59:20
   StartTime=2025-02-17T12:59:21 EndTime=2025-02-24T12:59:21 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2025-02-17T12:59:21
Scheduler=Main
   Partition=parallel AllocNode:Sid=hpc-camins:2963479
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=jade-node1
   BatchHost=jade-node1
   NumNodes=1 NumCPUs=8 NumTasks=1 CPUs/Task=8 ReqB:S:C:T=0:0:*:*
   ReqTRES=cpu=8,mem=8000M,node=1,billing=8
   AllocTRES=cpu=8,mem=8000M,node=1,billing=8
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   MinCPUsNode=8 MinMemoryNode=8000M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/users/jack.sparr/PROJ/PRUEBA_11/prueba11
   WorkDir=/users/jack.sparr/PROJ/PRUEBA_11
   StdErr=/users/jack.sparr/PROJ/PRUEBA_11/slurm-219.out
   StdIn=/dev/null
   StdOut=/users/jack.sparr/PROJ/PRUEBA_11/slurm-219.out
   TresPerTask=cpu=8
```

5.4.4.- Deleting jobs from the queues

The command to cancel a job is **scancel**:

```
scancel <JOBID>
```

5.5. Scientific Software

Listed below are the scientific programs installed in JADE and the basic user manual to send jobs to the queues.

5.5.1.- ABAQUS (and Tosca)

Abaqus is a calculation program for finite elements analysis. It is oriented to solve problems such as solid mechanics, linear and nonlinear problems in static and dynamic ranges.

Abaqus/Standard and Abaqus/Explicit are available for the analysis and Abaqus/CAE for data pre-processing and post-processing. Abaqus licenses are limited, for that reason we encourage to make responsible and moderate use of them. It is recommended **not to send more than two jobs per user using Abaqus/Standard and Abaqus/Explicit**, and just **one job for Abaqus/CAE**. If a user exceeds these limits, the system administrator is allowed to delete one of them without any previous warning.

The payment for the Abaqus license is made jointly between the research groups and the School of Civil Engineering. The fee to be paid will be determined based on the usage during the past year.

VERSIONS

The only version installed is Abaqus 2025.

Available versions:

Abaqus version	Commands to use	Default version
Abaqus 2025	abaqus, abq2025	Yes

To check the available licenses at any time you can use the following command:

```
> abaqus-licenses
```

ABAQUS INTERACTIVE

To execute **Abaqus/CAE** and have a visualization or post processing of the obtained data, it is required a Graphical Acces (see section 2.2) to get the visualization window. Abaqus/CAE command:

```
> abaqus cae -mesa
```

ABAQUS SERIAL JOB

To send an Abaqus job to the **serial** queue is necessary to use a shell script like this:

```
#!/bin/bash
unset SLURM_GTIDS
abaqus job=job_name input=input_filename.inp interactive
```

Shell command:

```
> sbatch [-p serial] <abaqus_script_name>
```

ABAQUS PARALLEL JOB

To send to the **parallel** queue an Abaqus job that uses more than one core is necessary to use a shell script like this:

```
#!/bin/bash
#SBATCH --cpus-per-task=<NumThreads>
unset SLURM_GTIDS
abaqus job=name input=filename.inp cpus=<NumThreads> interactive
```

Shell command:

```
> sbatch -p parallel <abaqus_script_name>
```

TOSCA

To execute **Tosca** in graphical mode, it is required a Graphical Acces (see section 2.2) to get the visualization window and also a Java interpreter available that can be loaded using «module» command.

```
> module load java/11.0.20
> tosca_gui
```

5.5.2.- MATLAB

MATLAB is a mathematic software that offers its own programming language, M language, integrated with its development environment. A shell script is required to send jobs to the batch queues. This shell script must have the following format, where *program.m* would be the filename of the Matlab file to execute:

```
#!/bin/bash
#SBATCH --mem=8192 #To increase memory limit to 8GB
echo "Running on host: " `hostname`
echo "Starting MATLAB at `date`"

matlab -nodesktop -nodisplay -nosplash < program.m > program.out
echo "MATLAB run completed at `date`"
```

Shell command:

```
> sbatch -p serial <matlab_script_name>
```

LAUNCHING MATLAB PARALLEL JOB WITH MULTITHREADS

To send to the **parallel** queue a Matlab job that uses multithreads is necessary to create a shell script specifying the number of threads:

```
#!/bin/bash
#SBATCH --mem=8192 #To increase memory limit to 8GB
#SBATCH --cpus-per-task=<number_of_threads>
echo "Running on host: " `hostname`
echo "Starting MATLAB at `date`"

matlab -nodesktop -nodisplay -nosplash < program.m > program.out
echo "MATLAB run completed at `date`"
```

Shell command:

```
> sbatch -p parallel <matlab_script_name>
```

LAUNCHING MATLAB EXPLICIT PARALLEL JOB

The Parallel Computing Toolbox is available on JADE. It is possible to use up to 48 workers for shared parallel operations on a single node in the current MATLAB version. Our license does not include MATLAB Distributed Computing Server. Therefore, multi-node parallel operations are not supported.

To send to the **parallel** queue a Matlab job that uses workers (**parpool** function) is necessary to create a shell script as shown below:

```
#!/bin/bash
#SBATCH --mem=8192 #To increase memory limit to 8GB
#SBATCH --ntasks-per-node=6
echo "Running on host: " `hostname`
echo "Starting MATLAB at `date`"
```

```
matlab -nodesktop -nodisplay -nosplash < program.m > program.out
echo "MATLAB run completed at `date`"
```

Shell command:

```
> sbatch -p parallel <matlab_script_name>
```

5.5.3.- OpenFOAM

OpenFOAM is the free, open source CFD software released and developed primarily by OpenCFD Ltd since 2004. It has a large user base across most areas of engineering and science, from both commercial and academic organisations. OpenFOAM has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to acoustics, solid mechanics and electromagnetics.

Available versions:

OpenFOAM version	Source configuration file	Default version
v2406	/usr/lib/openfoam/openfoam2406/etc/bashrc	No

We need to prepare OpenFOAM environment before to use it. It is necessary to source a script to set the needed variables. This is the example to load version v2406:

```
$ source /usr/lib/openfoam/openfoam2406/etc/bashrc
```

It is possible to configure an alias in your `.bashrc` to do it in a single command:

```
echo "alias of2406='source /usr/lib/openfoam/openfoam2406/etc/bashrc'"
>>$HOME/.bashrc
```

In this case, just login again and type `of2406` to be ready to use OpenFOAM.

Sending jobs to the queue

A shell script is required to send jobs to the batch queues. This shell script must have the following format:

```
#!/bin/bash
#SBATCH --mem=8192 #To increase memory limit to 8GB
echo "Running on host: " `hostname`
echo "Starting OpanFOAM at `date`"


source /usr/lib/openfoam/openfoam2406/etc/bashrc

cd <openfoam-working-directory i.e: /users/username/OpenFOAM/username-
v1812/run/pitzDaily/>
blockMesh > mesh.out #Use here your OpenFOAM needed commands
```

```
simpleFoam > output.out
echo "OpenFOAM run completed at `date`"
```

Shell command:

```
> sbatch -p serial <openfoam_script_name>
```

OpenFOAM can also be run in parallel, but needs further configuration. This is an example to send a job to the parallel queue:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=5
echo "Running on host: " `hostname`
echo "Starting OpanFOAM at `date`"

source /usr/lib/openfoam/openfoam2406/etc/bashrc

cd your_program_directory
mpirun -np 5 ./your_program_script > output.out
echo "OpenFOAM run completed at `date`"
```

Shell command:

```
> sbatch -p parallel <openfoam_script_name>
```

5.5.4.- OpenSees

OpenSees is a software framework for developing applications to simulate the performance of structural and geotechnical systems subjected to earthquakes.

OpenSees is available to run serial jobs (OpenSees executable) and parallel jobs (OpenSeesSP and OpenSeesMP executables).

In order to run OpenSees it is necessary to load the needed module first:

```
module load OpenSees/3.7.0
```

OpenSees version	Module
v3.7.0	OpenSees/3.7.0

Sending jobs to the queue

A shell script is required to send jobs to the batch queues.

<u>Serial Jobs:</u>

This shell script must have the following format:

```
#!/bin/bash
#SBATCH --mem=4096 #Default memory
#SBATCH --partition=serial
#SBATCH --job-name=test-opensees
#SBATCH --output=test_serial.log
#SBATCH --cpus-per-task=1 # OMP programs
date

module load OpenSees/3.7.0

OpenSees main.tcl
date
```

Shell command:

```
> sbatch -p serial <opensees_script_name>
```

Parallel Jobs: This shell script must have the following format:

```
#!/bin/bash
#SBATCH --mem=4096 #Default memory
#SBATCH --partition=parallel
#SBATCH --job-name=test-openseesMP
#SBATCH --output=test_parallel.log
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=16
date
module load OpenSees/3.7.0

mpirun OpenSeesMP main.tcl

date
```

Shell command:

```
> sbatch -p serial <opensees_script_name>
```

manual, jade, calculintensiu, PDI