

Environment and Modules

Environment Customization

After logging in, you may want to configure the environment. Write your preferred path definitions, aliases, functions and module loads in the .bashrc file

```
# ./bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
alias qs='qstat -a'
module load intel/2015b

# Display informations to standard output - only in interactive ssh session
if [ -n "$SSH_TTY" ]
then
    module list # Display loaded modules
fi
```

Do not run commands outputting to standard output (echo, module list, etc) in .bashrc for non-interactive SSH sessions. It breaks fundamental functionality (scp, PBS) of your account! Take care for SSH session interactivity for such commands as id="result_box" class="hps alt-edited">stated in the previous example.

Application Modules

In order to configure your shell for running particular application on Salomon we use Module package interface.

Application modules on Salomon cluster are built using EasyBuild. The modules are divided into the following structure:

```
base: Default module class
bio: Bioinformatics, biology and biomedical
cae: Computer Aided Engineering (incl. CFD)
chem: Chemistry, Computational Chemistry and Quantum Chemistry
compiler: Compilers
data: Data management & processing tools
debugger: Debuggers
devel: Development tools
```

geo: Earth Sciences
ide: Integrated Development Environments (e.g. editors)
lang: Languages and programming aids
lib: General purpose libraries
math: High-level mathematical software
mpi: MPI stacks
numlib: Numerical Libraries
perf: Performance tools
phys: Physics and physical systems simulations
system: System utilities (e.g. highly depending on system OS and hardware)
toolchain: EasyBuild toolchains
tools: General purpose tools
vis: Visualization, plotting, documentation and typesetting

The modules set up the application paths, library paths and environment variables for running particular application.

The modules may be loaded, unloaded and switched, according to momentary needs.

To check available modules use

```
$ module avail
```

To load a module, for example the OpenMPI module use

```
$ module load OpenMPI
```

loading the OpenMPI module will set up paths and environment variables of your active shell such that you are ready to run the OpenMPI software

To check loaded modules use

```
$ module list
```

To unload a module, for example the OpenMPI module use

```
$ module unload OpenMPI
```

Learn more on modules by reading the module man page

```
$ man module
```

EasyBuild Toolchains

As we wrote earlier, we are using EasyBuild for automatised software installation and module creation.

EasyBuild employs so-called **compiler toolchains** or, simply toolchains for short, which are a major concept in handling the build and installation processes.

A typical toolchain consists of one or more compilers, usually put together with some libraries for specific functionality, e.g., for using an MPI stack for distributed computing, or which provide optimized routines for commonly used math operations, e.g., the well-known BLAS/LAPACK APIs for linear algebra routines.

For each software package being built, the toolchain to be used must be specified in some way.

The EasyBuild framework prepares the build environment for the different toolchain components, by loading their respective modules and defining environment variables to specify compiler commands (e.g., via `$F90`), compiler and linker options (e.g., via `$CFLAGS` and `$LDFLAGS`), the list of library names to supply to the linker (via `$LIBS`), etc. This enables making easyblocks largely toolchain-agnostic since they can simply rely on these environment variables; that is, unless they need to be aware of, for example, the particular compiler being used to determine the build configuration options.

Recent releases of EasyBuild include out-of-the-box toolchain support for:

- various compilers, including GCC, Intel, Clang, CUDA
- common MPI libraries, such as Intel MPI, MPICH, MVAPICH2, OpenMPI
- various numerical libraries, including ATLAS, Intel MKL, OpenBLAS, ScaLAPACK, FFTW

On Salomon, we have currently following toolchains installed:

Toolchain	Module(s)
GCC	GCC
ictce	icc, ifort, imkl, impi
intel	GCC, icc, ifort, imkl, impi
gomp	GCC, OpenMPI
goof	BLACS, FFTW, GCC, OpenBLAS, OpenMPI, ScaLAPACK
iomp	OpenMPI, icc, ifort
iccifort	icc, ifort