Installation Manual

Amsterdam Modeling Suite 2026.1

www.scm.com

Apr 02, 2026

CONTENTS

CHAPTER

ONE

WINDOWS QUICKSTART GUIDE

1.1 Step 1: Download

Download AMS2026 for Windows (https://www.scm.com/support/downloads/) and save it to your Downloads folder.

Using Microsoft Edge? If you get a warning that the AMS installer is not commonly downloaded, select the "Keep" option, and "Keep anyway".

1.2 Step 2: Run the installer

To install AMS on Windows, you need Administrator privileges.

1. After the download is complete, open your Downloads folder in Windows Explorer and **double-click on the ams2026.101.pc64_windows.msmpi.exe** file. Newer versions of AMS will have a different number in the file name.

   Windows Smartscreen Defender may warn about an unrecognized app. You will need to press "Run anyway" in order to proceed with installation.

2. **Follow the on-screen instructions** to install AMS2026 to your computer. The InstallShield Wizard asks

   - where to install the program (`C:\AMS2026.101` by default),
   - where to save your AMS files (`C:\ADF_DATA` by default), and
   - where to store temporary data (`C:\SCMTMP` by default).

   Recommendation: Keep the default values. The paths can be changed, but should **NOT** contain any spaces!

3. During installation a **text console window** will open to extract a few more files.

   Do not close this window! If you do, the installation has to be started all over again.

Windows Firewall may ask you to grant network access to some executables. This permission is optional.

For third-party firewall applications you will need to grant access from scmd.exe to the localhost.

1.3 Step 3: Launch AMSjobs

Once the installation is complete, **double-click the "AMSjobs" shortcut** on the desktop to start AMS2026.

1.4 Step 4: Provide license information

A window may pop up asking for your license details. Enter your e-mail, SCM user id, and password. This will typically automatically create a valid license file `license.txt` in the AMS installation directory.

Problems? See how to *resolve license issues* (page 18).

Did you receive a license.txt file from SCM by e-mail or download? Put this file in the AMS installation directory from *step 2* (page 1), for example `C:\AMS2026.101`

1.5 Step 5: Tutorials

From AMSjobs, you can launch other AMS modules, like AMSinput (SCM → New Input) to set up new AMS calculations.

See the Getting Started tutorials to run your first AMS calculations.

1.6 Next steps

Tutorials Step-by-step guides from beginner to advanced.

Package Manager Additional downloads for ML Potentials, COSMO-RS compound database, …

Complete installation guide More detailed installation information.

Scripting *Python* and *bash* environments to automate your calculations.

License troubleshooting More information about how to solve license issues.

Support Get support through the SCM helpdesk.

(https://www.scm.com/contact-us/)

CHAPTER

TWO

LINUX QUICKSTART GUIDE

This quickstart guide is for installing AMS on a Linux desktop/laptop machine.

For cluster installations please read the *generic Installation manual* (page 15)

NOTE: The following steps should be taken under a normal user account. Do not use the root account or superuser!

2.1 Step 1: Download

Start with downloading AMS2026 for Linux from the main download page (https://www.scm.com/support/downloads/), and save it in your Downloads folder. You do not need to open the file.

Tip: Download the `IntelMPI (default)` version, if you are unsure which version to download. The IntelMPI runtime is included with the download.

If you have an AMD Zen processor (Ryzen/EPYC), then download the "Linux Intel MPI, optimized for AMD-Zen" binary instead.

2.2 Step 2: Extract the archive

Now **open a terminal** (Ctrl+Alt+T usually works, otherwise browse your application menus), and run the commands below to **extract the download into your home folder**.

Make sure to replace the ams2026.101.pc64_linux.intelmpi part to match the name of the file you downloaded (in case of a newer version). Use copy and paste (right-click with your mouse in the terminal screen to paste) to avoid mistyping the commands.

```
cd $HOME
tar -xf $HOME/Downloads/ams2026.101.pc64_linux.intelmpi.bin.tgz
```

2.3 Step 3: Set up the environment, source the amsbashrc.sh file

Run the following command to **source the amsbashrc.sh file**. Do not forget the dot and space at the beginning of the line! Also make sure to replace 2025.102 with the version you downloaded.

```
. $HOME/ams2026.101/amsbashrc.sh
```

2.4 Step 4: Start the AMSjobs graphical user interface

Start up the AMSjobs graphical user interface (GUI) with this command:

```
amsjobs &
```

If this fails to launch a GUI window, then you can try to run our GUI in software mode:

```
export SCM_OPENGL_SOFTWARE=1
amsjobs &
```

If this still does not open a GUI, then use your mouse to select all the text in the terminal window, copy it (right-click and select copy), and send us an email at support@scm.com with the text. DO NOT PROCEED WITH THE NEXT STEP!

2.5 Step 5: Install desired optional features with the package manager

From the GUI, navigate to **SCM -> Packages**.

- If you want to install for all users, hit the “User Mode” to switch to “Admin Mode”
- Select the features you’d like to install or select all with ctrl+A and click Install
- Wait, and close the package manager when done.

Alternatively, from the command line, make sure you’ve already installed a license and sourced amsbashrc.sh, then run the command:

```
# Install all available packages into AMSHOME, for all users:
"$AMSBIN/amspackages" --admin install all
```

2.6 Step 6: Create desktop icon

If the AMS GUI started without problems, go back to the terminal window and run the following command to **create a desktop icon** for AMS:

```
$AMSBIN/create_linux_icon.sh
```

2.7 Step 7: Automatically set up environment for all sessions

Finally we can set up our terminal to automatically source the amsbashrc.sh file when starting. **Add the source command to the .bashrc file** with the following command in the terminal window:

```
echo '. $HOME/ams2026.101/amsbashrc.sh' >> $HOME/.bashrc
```

You can also open the .bashrc file in a text editor, and manually paste the part between the quotes on a new line at the end of the file.

CHAPTER

THREE

MACOS QUICKSTART GUIDE

This quickstart guide is for installing AMS on an Apple MacOS machine. Please also read the *generic Installation manual* (page 15) if you encounter problems.

Start with downloading AMS2026 for MacOS from the main download page (https://www.scm.com/support/downloads/), and save it on your computer. Open ams2026.101.macarm64.openmpi.dmg after the download is complete. Newer versions of AMS will have a different number in the file name.

Important: AMS2024 is the last version of AMS to support the x86-64 Intel macOS platform. You can find AMS2024 in the "Previous Releases" section on our download page.

Drag the AMS application (called AMS20XX) from the disk image to your disk, for example by dropping it on the icon of the Applications folder in the disk image.

Important: the folder in which you store the AMS application should not contain spaces in its name (or in the names of any of its parent folders)!

Installing in the standard /Applications folder should work fine.

The AMS package needs a valid license file to run.

Double click the AMS20XX application.

When you open AMS20XX for the first time from your Applications, you'll see a notice "AMS20XX can't be opened because Apple cannot check it for malicious software". To Fix this

- In the Finder on your Mac, locate the app you want to open.
- Control-click the app icon, then choose Open from the shortcut menu.
- Click Open.

The app is then saved as an exception, and you can open it in the future by double-clicking it or open it from your Applications on the Dock.

If you have no valid license a window will appear that allows you to request and install a license file. If a license file is available for you it will be installed automatically.

Otherwise you will receive a mail when a license file is available. Double click the AMS20XX application again and request a license again to install it. Note that you will never get more than one license for a particular machine, so no need to worry about requesting a license twice.

If you want to install optional AMS components from the GUI, open **AMSjobs** and navigate to **SCM -> Packages**. You can then search for the packages you need, install them, and check whether updates are available.

For the full package manager instructions, see *Package Manager: Installing Optional Components* (page 31).

CHAPTER

FOUR

INTRODUCTION

This document describes the installation of the Amsterdam Modeling Suite (AMS) on the supported platforms. For optimal performance, some system specific tuning may be needed. Therefore, it is strongly recommended to also read the additional information and known issues .

The Amsterdam Modeling Suite consists of the following main classes of programs:

- Computational engines: ADF, BAND, COSMO-RS, DFTB, UFF, ReaxFF and MOPAC. Each of the engines has its own (text-based) input and output and can be used from scripts and the command line. New since the AMS2019 release is the AMS driver program, which houses the ADF, BAND, DFTB, UFF, MOPAC and ReaxFF engines.
- Utilities and property programs. These are used primarily for pre- and post-processing data of the computational engines.
- Graphical user interface (GUI), which is used to prepare input for computational engines and to visually present their results.

AMS is bundled as a mostly complete package and all engines are installed at once. Your license file determines which of the functionality will be available after installation. Some optional functionality can be installed afterwards, see *Package Manager: Installing Optional Components* (page 31).

The AMS package is written with a Unix-like environment in mind, but Unix/commandline knowledge is not needed install or use AMS from the GUI. Linux laptop and desktop users can follow the instructions in the *Linux Quickstart Guide* (page 3) to install AMS. Windows users can follow the on-screen instructions after starting the installer, or take a look at the *Windows Quickstart Guide* (page 1). MacOS/OSX users can simply drag&drop the AMS application to install it, see the *MacOS Quickstart Guide* (page 7) for more details.

If you plan to do advanced scripting or run AMS from the command line, then you will need to know how to write shell scripts and have some knowledge of the environment variables. If you are the one who is going to install the package on a shared Unix-like system, such as Linux cluster environment, then you need to know how to modify shell resource files such as .bashrc.

4.1 Requirements

4.1.1 Summary

AMS2026 can be used on anything from a simple laptop to big Linux cluster environments, and is tested on a wide variety of hardware.

Recommended minimum hardware specification:

- Intel i5 or better 64bit CPU supporting AVX2 instructions
- 8GB RAM

- 250GB SSD
- OpenGL 3.2 graphics

Absolute minimum hardware specification:

- 64bit CPU supporting AVX2 instructions
- 2GB RAM
- 6GB storage for installation
- OpenGL 1.4 graphics

Note that for macOS, we only support Apple Silicon M1 processors and newer, not Intel. On Windows and Linux ARM64 CPUs such as Qualcomm Snapdragon are not supported at this time.

Supported Operating Systems:

- Windows 10/11
- macOS 13.2 or newer
- Linux with GLIBC v2.28 or higher, including: Rocky8/Alma8/RHEL8 or newer, Debian 10 or newer, SUSE 15.6 or newer, Ubuntu 20.04 or newer, etc. AMS is regularly tested on Almalinux 8, Ubuntu 24.04 and Arch Linux.

Specific hardware and software requirements for the AMS package depend on the platform. The list of supported platforms, including information on the operating system, parallel environment (MPI), and compilers used is available in the Download section (https://www.scm.com/support/downloads/) of our web site.

4.1.2 Detailed Hardware requirements

CPU

AMS2026 works on modern Intel CPUs with AVX2 or AVX-512 instruction sets and AMD Zen CPUs (Ryzen, Threadripper, EPYC). Especially the AMD Zen2/Zen3 processors give very good performance (Ryzen 3000/4000/5000, EPYC 7**2/7**3). The macOS version of AMS2026 works on Apple Silicon (arm64) processors. We no longer provide a binary for older intel (x86) macOS systems. Note that AMS for Windows and Linux does not work on arm64 processors, such as Qualcomm Snapdragon.

Memory

In a parallel calculation, the total amount of memory used by the job is a sum of that used by each process. Starting from ADF2010, some large chunks of data are placed in the shared memory so the sum rule does not strictly hold. In principle, it is more correct to count memory per process but since AMS is an MPI program it runs most efficiently when the number of processes corresponds to the number of physical processors cores. Therefore, all memory amounts below are per processor core.

The amount of RAM per core needed to run AMS depends greatly on the kind of calculation you perform. For small calculations, 256 MB will be sufficient, but if there is a lot of memory available AMS may run significantly faster. A large amount memory also reduces the load on the disks, which may speed up your calculation depending on the I/O sub-system and the operating system.

The memory requirement increases with the system size. For example, for a molecule containing 100 atoms with a DZ basis set it may be sufficient to have 256 MB but for a molecule with 1000 atoms up to 8 gigabytes may be required. Also, if you are going to perform TDDFT, relativistic spin-orbit or analytical frequency calculations then the amount of memory should be larger. As an indication, an analytical vibrational frequency calculation of a organometallic complex containing 105 atoms with a TZP basis set uses up to 1GB of RAM per process but it can be done with less, even though not as efficiently.

Disk

For installation of the package on Linux/Unix you need from about 5GB (without sources) to 8GB (with sources and compiled objects). The run-time (scratch) disk space requirements greatly depend on what type of calculation you perform. For the ADF engine, it may range from a few megabytes for a small molecule up to a hundred gigabytes for a large molecule with a large basis set, the amount scaling quadratically with the system size. Using a Solid State Drive (SSD) helps performance, especially for bigger calculations.

Network

First of all, a network card must be present in the computer as its hardware MAC address is used as the computer's ID for the licensing.

In order to enable MPI on a standalone Windows computer, one may need to create a dummy network connection by adding a network "card" called Microsoft Loopback Adapter. This interface will be assigned an IP address from a private range.

Multi-host parallel performance.

As far as performance concerned, a switched Gigabit Ethernet network is typically sufficient for good parallel performance on up to four nodes if the nodes do not have too many CPU cores per node. If you are going to use more nodes, or nodes with high core count (>16 cores per node), you may need faster communication hardware, such as Infiniband, to get good performance. Please note that multi-host execution is not supported on Windows.

Graphics

The AMS GUI requires OpenGL 3.2 to run. For Linux and Windows users there is an OpenGL software mode available for older hardware, please read more about it in the Remote GUI documentation.

4.1.3 Software requirements

Operating System

The package runs on Windows and on the following Unix variants: Linux, Mac OS.

On the Apple systems the macOS 13.2 (Ventura) and newer is supported on Apple Silicon hardware.

On linux both the compute engines, python scripting environment and GUI require a GLIBC version of 2.28 or higher. The main binaries of AMS are compiled on CentOS 7, the code gets tested daily on Almalinux 8, Ubuntu 22.04 and Arch.

The Windows version of AMS is supported on the desktop editions of Windows (10 and 11). The Windows Server is **not** supported.

Additional libraries

Certain version of AMS will require different libraries to be installed on the system depending on the MPI library used.

Graphics

In order to run the the graphical user interface (GUI) the computer needs to have an OpenGL-capable graphics subsystem (hardware, drivers and libraries). Besides that, on Linux the following (or equivalent) packages must be installed:

```
fontconfig
freetype
libdrm
libICE
libSM
libstdc++
libcrypt (part of glibc on many modern systems)
libX11
libXau
libXdmcp
```

(continues on next page)

(continued from previous page)

```
libXext
libXft
libxkbcommon-x11
libXrender
libXScrnSaver (Ubuntu users may need to install libXss1)
libXt
libXxf86vm
mesa-libGL
mesa-libGLU
```

The GUI will not able to start without shared libraries provided by these packages.

NOTE: If you receive an error about libXss (libXss.so.1: cannot open shared object file: No such file or directory), you need to install libXScrnSaver (redhat/centos: yum install libXScrnSaver) or libxss1 (ubuntu/debian: sudo apt install libxss1).

Compiler

If you have a license for the source code, you can compile the source yourself, with or without your own modifications.

The source consists mainly of Fortran95/2003 code, with some small parts written in C. Some of the Fortran2003 features are also used so a compiler supporting it is required. You must use object-compatible Fortran and C compilers to those we are using on the same platform, since some parts of the code are available only as object modules. For all x86 platforms it is currently Intel OneAPI Fortran 2024.1.0 (also labeled 2021.12.0). It is very unlikely that other compilers, or even a different major version of the same compiler, will work with these object modules. We cannot support compilers different from what we are using ourselves.

To check which compiler to use, see *Platform Specific Information* (page 97).

4.2 Important changes since AMS2025

- No more Centos 7 support: The minimal glibc version requirement for AMS on linux is 2.28.
- Package manager installation procedure for administrators has been simplified, defaulting to AMSHOME.

4.3 Important changes since AMS2024

- Intel x86 macOS is no longer supported.

4.4 Important changes since AMS2023

- AMS2024 will be the last version of AMS to support Intel x86 macs.
- There is only one Windows version of AMS, using Microsoft MPI and Intel MKL. This version should work both on Intel and AMD machines.

4.5 Important changes since AMS2022

- No more Centos 6 support: The minimal glibc version requirement for AMS on linux is 2.17.
- The AMD optimized Linux binaries now use AOCL (https://www.amd.com/en/developer/aocl.html), instead of MKL, for better performance on AMD Zen-based processors.
- The AMS Python Stack has been updated to the Enthought Python v3.8.12 stack, along with updates of several of the included packages.
- The Windows versions of AMS now use Microsoft MPI (https://learn.microsoft.com/en-us/message-passing-interface/microsoft-mpi) for parallelization, instead of intel MPI.

4.6 Important changes since AMS2021

- The start script has been made more flexible, and most often modifications to it are no longer needed. The MPI executable can now be supplied via the `SCM_MPIRUN_EXE` variable, and arguments to the MPI startup can be supplied via the `SCM_MPIRUN_OPTIONS` variable. The autodetection mechanism for cluster queues is working as before. If you only want to supply additional MPI arguments and not overwrite the ones automatically selected by AMS, use the `SCM_MPIOPTIONS` variable. See *Appendix A. Environment Variables* (page 71) for more details.

4.7 Important changes since AMS2020

- AMS now has a package manager for installing optional components. For more details see Installing Optional Components
- Linux binaries for AMD Zen processors now use MKL instead of OpenBLAS. These binaries are forced to run in AVX2 mode, and give better performance on AMD Zen-based processors.

4.8 Important changes since AMS2018

No major technical changes have been made in AMS2019

4.9 Important changes since ADF2017

Some of the technical changes made since ADF2017:

- No more support for 32-bit Windows: almost all PCs nowadays run on 64-bit processors and 64-bit Operating Systems. If you need a 32-bit version, we advise you to use ADF2017 instead.
- Automated OpenGL fallback mode for GUI on older graphics: available on 64-bit Windows and Linux.
- No more support for CentOS 5 on Linux: The Intel ifort 18 compiler unfortunately introduces an unavoidable GLIBC 2.11 requirement, which means AMS2018/AMS2019 no longer works on older Linux systems.
- Linux binaries for AMD Zen processors: AMS2018/AMS2019 is available with OpenBLAS instead of MKL, optimized for the AMD Zen architecture. These binaries should be used on AMD Ryzen / Threadripper and Eypc CPUs for better performance.
- Updated Python stack: Our python stack is now based on the Enthought Python v3.6 stack, most included modules have also been updated.

- AVX-512 support: AMS2018/AMS2019 is compiled with AVX-512 optimizations for the latest Intel Xeon Scalable Processors (Skylake SP).

CHAPTER

FIVE

INSTALLATION

Typically installation of the AMS package is simple and straightforward. If you have problems installing it, create a support ticket (https://support.scm.com/portal/en/signin) and provide as much information as possible so that we can assist you.

To install the AMS package you have to go through the following steps:

Below we discuss each step separately.

5.1 Decide which version to install

Choose the released version or a snapshot.

- Normally, you will install the released version from the main download page (https://www.scm.com/support/downloads/).
- The bug-fixed binaries (https://www.scm.com/downloads/#binaries) contains the most recent bug fixes. You may want to install a snapshot version if you know that some bugs have been fixed recently.
- The development snapshots page (https://www.scm.com/downloads/#binaries) contains the latest developments. You will only want to download it if you need to use the latest functionality not available in AMS2026.

Choose a platform. AMS is available for a number of platforms. A platform is a combination of the processor architecture, operating system, MPI implementation, and any other specialties such as CUDA or "optimized for AMD-Zen". AMS2026 is only supported on 64-bit Operating Systems (and hardware).

Currently, the following platforms are officially supported and available from our website: + Linux: x86-64 (64-bit): IntelMPI, OpenMPI, IntelMPI, IntelMPI optimized for AMD-Zen (Optimized for Ryzen/Threadripper/EPYC processors) + macOS: Apple Silicon (ARM64) with OpenMPI + Windows: x86-64 (64-bit) with Microsoft MPI

On Windows and Linux ARM64 CPUs such as Qualcomm Snapdragon are not supported at this time.

Architecture on macOS. We support AMS on *Apple Silicon* macs natively on macOS Ventura and later (13.2+). Using intel binaries of older AMS versions on Apple Silicon macs is not supported. If you receive an `Illegal instruction` error, you downloaded the wrong version (*FAQ* (page 100)).

Choose MPI (on Linux). We advise to use the IntelMPI version if possible. It has been tested on both desktop and cluster machines and should work out-of-the-box on most cluster batch systems. The IntelMPI runtime environment is distributed with AMS. You do **not** need to install it. If you prefer to use the OpenMPI version instead, keep in mind that if you wish to run multi-node calculations you need to use a local copy of OpenMPI 4.1.1 with support for your batch system build in. The OpenMPI runtime environment is distributed with AMS but it is limited to single-node (or desktop) usage.

Cray XC. Cray users can use the IntelMPI version of AMS because it is binary-compatible with Cray MPI shared libraries. Read the MPICH ABI compatibility section in the Cray documentation for more information, and talk to your local system administrator. If the machine has a "cray-mpich-abi" module available, you can try using that in combination with the SCM_USE_LOCAL_IMPI=1 environment setting.

GPU acceleration: **note: ADF in AMS2026 does not offer GPU acceleration support** We don't offer a GPU-accelerated version of ADF at this moment in time. However, some of the machine learning potentials available in AMS can benefit from GPU acceleration by installing the appropriate version of the potential from the SCM package manager.

Optimized for AMD-Zen: AMS2018 introduced a new supported platform, optimized for AMD Zen (Ryzen/Threadripper/EPYC) processors. Initially this used the OpenBLAS library, but now this platform also uses Intel MKL on Windows. This version will NOT work on older AMD processors. Starting with AMS2023 we offer AMD binaries using AOCL on Linux, instead of MKL.

5.2 Download and install the software

All systems: Download an installation package from one of the download pages mentioned above.

Linux: The installation package is a compressed tar archive. Untar it in the directory of your choice. Desktop users can follow the Linux Quickstart Guide to install AMS2026. Please note that in a cluster environment AMS must be installed on a shared file system accessible from all nodes of the cluster. The archive contains one directory, ams2026.10x, which, when installed, will be referred to as *$AMSHOME*. The installation package has the IntelMPI runtime and Intel MKL libraries included, you do not need to install these separately!

macOS: The installation package is a disk image (.dmg) file. To install AMS on macOS, double click on the downloaded disk image and drag *AMS2026.10x* somewhere on your hard disk, such as the *Applications* directory. Be sure to **read the enclosed ReadMe** file for further important details!

Windows: The downloaded file is an executable Microsoft Installer package containing an installation wizard that will guide you through the installation process. If you receive a Windows Smartscreen Defender warning after opening the file, please select "Run anyway" to proceed. Open the file after downloading, and follow the instructions. **note: Do not close the console window that pops up during the installation!** . Please note that you need Administrator privileges to install AMS. After or during the installer you might be asked by the Windows Firewall to grant network access to some executables, this permission is optional. For third-party firewall applications you will need to grant access from scmd.exe to the localhost.

5.3 Set up the environment

Windows users can skip to the next section since for them the environment is set up by the installation wizard.

macOS users may follow the UNIX instructions if they (also) wish to run from the command line. However, read the ReadMe file inside the dmg file for some important extra details! Alternatively, Mac users can start by double clicking the AMS2026.10x application. The AMS2026.10x application will automatically set the environment, and start AMSjobs for you. Next, all tasks (GUI or computational engines) started from this AMSjobs will inherit the proper environment.

For users of any **Unix-like** system the following step is mandatory.

For a **single-user** installation, the end-user is likely also the person installing AMS. If the user has a bash shell, it should be sufficient to source the $AMSHOME/amsbashrc.sh:

```
. $HOME/ams2026.10x/amsbashrc.sh
```

Z shell users can source

```
. $HOME/ams2026.10x/Utils/amszshrc.sh
```

Alternatively, it is also possible to **edit** the $AMSHOME/Utils/amsrc.sh (for sh/bash/zsh users) or $AMSHOME/Utils/amsrc.csh (for csh/tcsh users) file to set a number of important environment variables and source the file:

```
. $HOME/ams2026.10x/Utils/amsrc.sh
```

or (for tcsh)

```
source $HOME/ams2026.10x/Utils/amsrc.csh
```

Note: If the amsrc.sh or amsrc.csh file is missing from your Utils folder, you can download then here: `Download amsrc.sh` , `Download amsrc.csh`

To set up the environment automatically when starting a new terminal, add the source command to the $HOME/.bashrc file. It is also possible to create a launcher icon for the GUI by running the $AMSBIN/create_linux_icon.sh script AFTER the environment has been set:

```
$AMSBIN/create_linux_icon.sh
```

For a **multi-user** installation, you can either copy both amsrc.* files to the /etc/profile.d directory (after editing them) or, if your system supports modules, create a module file for AMS2026. The following environment variables must be defined in the module file:

- AMSHOME: AMS installation directory
- AMSBIN: should be equal to $AMSHOME/bin
- AMSRESOURCES: should be equal to $AMSHOME/atomicdata
- SCMLICENSE: complete path of the license file, typically $AMSHOME/license.txt
- SCM_TMPDIR: path to the user's scratch directory, for example, /scratch/$USER. This directory must exist prior to the first execution of any program from the AMS package by that user. Thus it may be a good idea to add to the user's profile a command that creates the directory in case it does not exist. You can also choose to use the same directory for all users. See SCM_TMPDIR Environment variable for more details.
- SCM_PYTHONDIR: it should point to the location where you want the SCM python stack to set up the virtual environment. This is for installing additional python packages from PyPi with "amspython -m pip install" and with the package manager. Typically SCM_PYTHONDIR=$HOME/.scm/python on Linux.

A complete list of environment variables is provided in Appendix A.

If you are planning on using the GUI on a remote machine (for example via ssh with X-forwarding), make sure to also take a look at Using the GUI on a remote machine.

If you plan on running jobs directly from the command line, please do **not** mpirun the programs directly. The MPI startup is done automatically, for more information see the Additional Information on running MPI jobs.

5.4 Set up the license

Which functionality can be used on each computer is determined by the license file pointed to by the SCMLICENSE environment variable.

When you start any GUI program from the AMS package (like ADF, BAND, AMSjobs or AMSinput) it will try to install a license file for you if necessary. To do this, some information is needed:

e-mail:
Your e-mail address, this may be used to contact you and send you a license file.

SCM userid (uxxxxx):
The same as the download login you should have received.

Password:
The download password belonging to the SCM userid

Remember:
If checked, the SCM userid and password will be saved on your computer, so you do not have to enter them again. They are saved in a readable format, so only do this on a computer and account you trust.

Get license without user confirmation:
If checked, a license request will automatically be issued when necessary, without user intervention. The intended use is for running on clusters, or for use in classroom/workshop situations. It will work only after arranging this

with SCM first.

Click OK to request and install a license to run on your current machine. The information will be sent to the SCM license server.

The same autolicense procedure can also be started from a shell environment:

```
$AMSBIN/autolicense nogui -u username -p password -m mailaddress
```

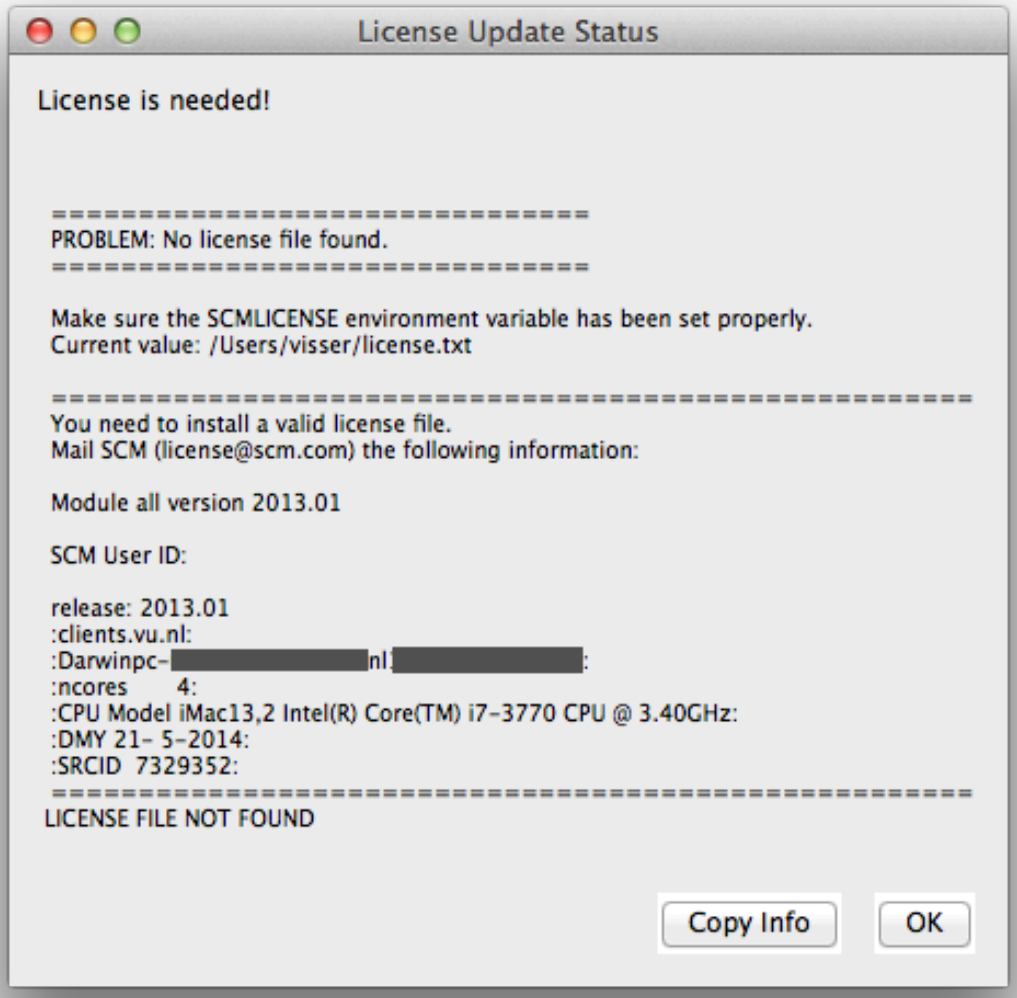
If a license is available, or can be generated by the license server (for **trial** users) it will be installed automatically. Obviously you will need to have an internet connection with access to the outside world for this to work.

For **Non-trial** users, license requests are handled manually. After some time (between hours and days) you will be notified by mail that a license file has been prepared for you.

Run the software again (on the same machine with the same SCM userid), and in many cases the license will automatically be installed for you. If not, follow the "Manual license installation" instructions (further down on this page).

Click Cancel when you do not want to request a license to run on the current machine, if your machine has no internet connection, or if you wish to request and install a license file manually.

A window will appear (either a regular Text editor on your system, or the (License Update Status) appears telling you a license is needed:



It will show the info that needs to be mailed to SCM to get a license. You can copy that information to your clipboard by clicking the Copy Info button, or the usual copy tool in your editor.

Next, mail this information to license@scm.com if you indeed wish to request a license for this machine.

After some time (hours-days as the license request is processed manually) you should receive a mail from SCM containing the license file.

You have receive a new license file via email. **Do not make any changes to it**, as that will break the license!

Mac users can normally just drop the license file on the AMS2026 application icon.

Other users should save the license file such that $SCMLICENSE points to it. Note the value of SCMLICENSE was shown in the License Update Status dialogue.

The default location of the license file (for Windows and the default SCMLICENSE value from a adfrc.* file) is set to $AMSHOME/license.txt, which means you should save the file as license.txt in the AMS installation directory. For macs the default location is "$HOME/Library/Application Support/SCM/license.txt".

Unix-like users can generate the license information by running the following command at the shell prompt in a terminal window:

```
$AMSBIN/dirac info
```

Output of this command from all computers on which AMS will be running, including all nodes of a cluster if applicable, must be sent to license@scm.com.

After receiving this information SCM will prepare a license file matching your license conditions and e-mail it to you with instructions on how to install it.

After receiving your license file you will need to save it so that $SCMLICENSE points to it.

In a multi-user environment, make sure that permissions on the license file allow read access for everyone who is going to run AMS.

5.4.1 Floating license

Note: floating licenses are only intended for Linux clusters.

If you have a floating license, you will need to follow these instructions below to set it up. The instructions are simple, but it is important you follow them exactly.

If you do not have a floating license you may skip this section.

Create a floating license directory

Make a new directory, for example FloatADF, to keep track of the running ADF processes.

This FloatADF directory must be:

- in a fixed location (the directory name should not change!)
- shared between / mounted on all nodes where you want to run ADF
- writable by all users that want to run ADF
- FloatADF must **not** be a subdirectory of $AMSHOME

For example:

```
cd /usr/share
mkdir FloatADF
chmod 1777 FloatADF
```

If you (also) have a floating license for BAND, DFTB or ReaxFF, you need to similarly set up directories FloatBAND, FloatDFTB, FloatReaxFF. The permissions can also be slightly more restrictive, like 1770, if the folder is group-owned for a specific unix group of AMS users.

In the example, we have given all users read, write and execute permissions to this directory. If you wish to restrict this for security reasons, you may do so as long as all ADF users will have read, write and search permission for this directory. You may create an ADF user group for this purpose.

Important: the directory should **not** be a sub-directory of $AMSHOME as the directory name will change if you update to a new version! Also, do **not** put the license.txt file in the FloatADF directory.

The FloatADF directory may not be moved, deleted or renamed for the duration of your license because these actions will invalidate it!

E-mail us the license information Send the result of the following commands (using again the name FloatADF and the location /usr/share as an example) to license@scm.com:

```
cd /usr/share
ls -lid $PWD/FloatADF
```

Please note that output of the ls command must include the full path of the FloatADF directory.

Together with the output of the ls command above, you also need to send us output of the *$AMSBIN/dirac info* command from each computer on which ADF will possibly run, as the license file is still host-locked.

In the case of very large clusters, it is often sufficient if you send us the output of the *$AMSBIN/dirac info* command from the head node and 2 or 3 compute nodes. As most compute node names have the same beginning, we can then add them with a wild card (for example Linuxchem*). It is important when you use this facility, to let us know that the info you are sending are not all compute nodes. Otherwise the license will only run on these few compute nodes.

5.4.2 Cloud license

If you have a cloud license, export `SCM_CLOUD_CREDS` with your SCM credentials before running AMS:

```
export SCM_CLOUD_CREDS="<your credentials>"
```

AMS will then retrieve the license over the internet instead of reading a local `license.txt` file.

Check that the cloud license works for a specific target with `dirac TARGET`. For example, to check ADF run:

```
$AMSBIN/dirac ADF
```

Use the relevant target name for other licenses. If the command produces no output, the cloud license is working. Otherwise you will get an error such as:

```
server 1: Server response code: 401
server 2: Server response code: 401
server 3: Server response code: 401
Could not obtain license.
```

With further details, such as `Invalid username or password` or `Target not enabled in the cloud license`.

5.5 Set up the scratch space

Most programs from the ADF package use disk for temporary data. This data often takes a significant amount of space and is used frequently. To avoid run-time errors and performance loss you may want to make sure that the file system used for temporary files is both big and fast. The SCM_TMPDIR environment variable is used to tell the programs where to put their temporary data. Please note that SCM_TMPDIR should always be set. If it is not set then each process will create its own directory in the current working directory where is was started.

Please see Appendix A on additional information about the SCM_TMPDIR variable.

Using multiple disks

Sometimes, if you have multiple non-RAID disks in your system, you may want to spread scratch files across different physical disks for better performance. It is possible to request that every AMS MPI-rank creates its files in a different directory by adding "%d" in $SCM_TMPDIR. If a "%d" string is encountered in the value of SCM_TMPDIR variable it will be replaced by the MPI rank number of the process at run-time. This means, however, that you may have to create up to 128 or more (depending on the maximum number of processes in one parallel calculation) symbolic links on each node where AMS is supposed to run. You should also create a directory matching the SCM_TMPDIR's value literally so that any process that does not interpret '%d' could also run.

Example: suppose there are two scratch file systems, /scratch1 and /scratch2 located on two different physical disks of an 8-core workstation. We want the odd rank numbers to use /scratch2 and the even numbers to use /scratch1. One way to achieve this is to create an empty /scratch directory and create nine symlinks in it as follows:

```
ln -s /scratch1 /scratch/%d
ln -s /scratch1 /scratch/0
ln -s /scratch2 /scratch/1
ln -s /scratch1 /scratch/2
ln -s /scratch2 /scratch/3
ln -s /scratch1 /scratch/4
ln -s /scratch2 /scratch/5
ln -s /scratch1 /scratch/6
ln -s /scratch2 /scratch/7
```

After that set SCM_TMPDIR to "/scratch/%d" and the ranks 0, 2, 4, 6 will use /scratch1 while ranks 1, 3, 5, and 7 will use /scratch2. When running AMS on a cluster it is better to combine multiple disks in a RAID 0 (striping) configuration as you probably do not want to create hundreds of symbolic links on each node.

5.6 Test your installation

This is a very important step and it should never be skipped.

5.6.1 Check the license file

Windows users test their license by double-clicking the makelicinfo.bat file in the AMS installation folder.

Unix users: first check that the license file has been installed correctly by running (at the shell prompt):

```
$AMSBIN/dirac check
```

This should produce the output similar to the following:

```
 Checked: /home/testadf/ams2022.xxx/license.txt

  License termination date (mm/dd/yyyy):  1/ 8/2022

  According to it, you are allowed to run:
        ADF version 2022.990
       BAND version 2022.990
       DFTB version 2022.990
    AMSMOPAC version 2022.990
      DCDFTB version 2024.000
      QNDFTB version 2022.990
      REAXFF version 2022.990
       MLPOT version 2022.990
  FORCEFIELD version 2022.990
      ADFGUI version 2022.990
     BANDGUI version 2022.990
     DFTBGUI version 2022.990
   REAXFFGUI version 2022.990
    MOPACGUI version 2022.990
       QEGUI version 2022.990
         CRS version 2022.990
         GUI version 2022.990
```

(continues on next page)

(continued from previous page)

```
          NBO version 2022.990
         NBO6 version    6.000
          AMS version 2022.990
        Utils version 2022.990

   Number of procs you are allowed to use for:

ADF       :     128 procs
BAND      :     128 procs
DFTB      :     128 procs
MOPAC     :     128 procs
ReaxFF    :     128 procs
MLPOT     :     128 procs
FORCEFIELD:     128 procs
AMS       :     128 procs

   ======================================================
   SCM User ID: u999999
   release: 2022.101
   :example.com:
   :Linuxmaster.example.com00:11:22:33:44:55:
   :ncores      12:
   :CPU Model Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz:
   :DMY  1- 7-2022:
   :SRCID  3217288:
   ======================================================
  LICENSE INFO READY
```

If the license check is successful (i.e. you get the "LICENCE INFO READY" message) you can move on. Otherwise check that the license file has been installed according to instructions above.

5.6.2 Run some basic tests

Verify that you can now run examples provided with AMS and that they give correct results. We recommend that you consult the Examples document for notes on such comparisons: non-negligible differences do not necessarily indicate an error. If you have a license for the GUI you can also run a few GUI tutorials as a test.

Note: the example *.run files are complete Bourne shell scripts that should be executed as such, they are **not* input files to be fed into any program. The easiest way to run them is using AMSjobs.

5.6.3 Test the GUI

If the GUI is included in your license, check that you can start any of the GUI modules.

UNIX users:

Enter the following command:

```
$AMSBIN/amsjobs
```

An AMSjobs window should appear. If your GUI crashes, especially when starting AMSinput, try setting the SCM_OPENGL_SOFTWARE=1 environment variable before launching the GUI:

```
export SCM_OPENGL_SOFTWARE=1
$AMSBIN/amsjobs
```

Mac users:

Double click the AMS2026.10x application to start it. An AMSjobs window should appear.

Windows users:

Double click the AMSjobs icon to start it. An AMSjobs window should appear. If the window does not appear or appears after a long delay then check the AMS-GUI requirements and check the firewall rules that should allow local communication.

All users should now be able to start AMSinput via the SCM menu on the top left of the menu bar.

5.6.4 Test parallel execution

It is very important to make sure that computer resources are utilized with the maximum efficiency. Therefore, you should check that each AMS job uses all processors/cores allocated to it and that the network is not overloaded with the disk I/O traffic.

Typically, when you submit a parallel job to the batch system you have a possibility to specify how many processors per node (ppn) to use. If the batch system you are using allows this, then make sure that you request ppn equal to the number of physical cores on each node. Note that old AMD processors based on the so-called "Bulldozer" architecture have one floating-point unit (module) per two physical cores. On this architecture and its successors (Piledriver, Steamroller) AMS will probably perform best when you only run on half the physical cores. When running outside of a batch system AMS tries to detect the optimal number of processes to start automatically, but when running under a batch system it's the user's responsibility set the ppn to an appropriate value. This is no longer a problem with the AMD Zen (Ryzen/Threadripper/Epyc) architecture. However, for these processors it is advisable to use the "IntelMPI optimized for AMD-Zen" version of AMS2026. This version forces the Intel Math Kernel Library (MKL) to run in AVX2 mode for the best performance, because otherwise the auto-detection mechanism in MKL switches it to the slowest code-path for non-Intel processors.

It is also possible that your batch system does not allow you to specify ppn but instead it always assumes that there only one processor per node. In this case, you will need to edit the $AMSBIN/start file and add some commands for processing the hostfile.

In order to check that everything is working as intended, pick one of the jobs found in the examples/Benchmarks folder, depending on which functionality you are mostly interested in. For example, take the ADF/Si35_TZ2P for molecular DFT, BAND/GuanineRing2D_gga for periodic DFT, DFTB/ubiquitin for approximate DFT (DFTB), or Reaxff/AMS_PETN_LAMMPS for reactive force-field MD with AMS. Additional information for some of the examples can be found in the corresponding Readme.txt and/or timings.txt files, where available.

Copy the run file from the selected example to an empty directory and start it using amsjobs, preferably on more than one node. After the job has finished (which can take anywhere from a few minutes to hours, depending on the example), open the job's out file and find the table that looks like the following:

```
Parallel Execution: Process Information
================================================================================
Rank   Node Name                                NodeID   MyNodeRank  NodeMaster
   0   compute-0-0                                 0           0           0
   1   compute-0-0                                 0           1          -1
   2   compute-0-0                                 0           2          -1
   3   compute-0-0                                 0           3          -1
   4   compute-0-1                                 1           0           1
   5   compute-0-1                                 1           1          -1
   6   compute-0-1                                 1           2          -1
   7   compute-0-1                                 1           3          -1
================================================================================
```

Check the names in the "Node Name" column and verify that the number of tasks per node is as expected. If it is, then move on to the next section.

Please note that there may be more than one AMS node (distinguished by the NodeID) per physical node (distinguished by the Node Name) depending on the node's NUMA settings and the values of the SCM_SHAR_NCORES and SCM_SHAR_PER_SOCKET environment variables, see Environment Variables for details.

5.6.5 Test parallel performance

If the process allocation to nodes looks as expected then scroll down a bit to the *Temporary files are created in* line and make sure you do not see anything like *Temporary file I/O may be slow*. If you see it then you should consider setting the SCM_TMPDIR environment variable to a path on a local or a high-performance distributed file system. Do not point SCM_TMPDIR to a directory on an NFS file system because doing so may slow down your computations to a crawl!

Scroll down a bit more to the lines that look like this:

```
Communication costs MPI_COMM_WORLD:      1.026 usec per message,   0.0320 usec per 8-
↪byte item
Communication costs for intra-node:      1.023 usec per message,   0.0318 usec per 8-
↪byte item
```

Make sure that you get reasonable numbers (less than 100 usec per message) both for intra-node and MPI_COMM_WORLD communication costs. Otherwise contact your system administrator. High communication cost for intra-node may mean the CPU affinity is not configured correctly or the machine is oversubscribed (which is extremely bad). If only the MPI_COMM_WORLD communication cost is high but the intra-node one looks reasonable, this may mean that the network link between machines is very slow and you may want to run single-node jobs only.

If this is OK then scroll down to the "NORMAL TERMINATION" line. The basic timing statistics is printed after the line, which may look like this:

```
    Total cpu time:      58.28
 Total system time:       0.84
Total elapsed time:      59.72
```

This shows how much time (in seconds) was spent in the AMS code (cpu time) and in the kernel (system time), and how long did it take for the calculation to complete (elapsed time). Ideally, the system time should be small compared to the cpu time and the latter should be close to the elapsed time. The system time will not always be a small number but the total of the system and cpu time should always give a number very close to the elapsed time. If this is not the case then it means that AMS has to spend a lot of time waiting for something. This can be, for example, disk I/O or network communication.

If you notice that the system time portion is large or that there is a large difference between the cpu+system and the elapsed time, then send the output file to the SCM support. Alternatively, you can analyze the detailed timings table that follows the basic stats yourself. Make a note of the timers that contribute to the irregularities the most and compare them with the same table found in the example's directory. The timer name may provide a hint of what part is slow (for example, if you see something like 'read' or 'write' in the timer name then it's likely related to the disk I/O). If you have trouble interpreting the results then you can still send the file to SCM support.

5.7 Configure AMSjobs queues and 3rd party software (optional)

If you would like to use the GUI as a front-end to submit your jobs to queues, you should configure those queues in AMSjobs. Third party software MOPAC and ffmpeg (export movies) may also be installed separately.

5.7.1 AMSjobs queues

A video shows how to set up remote queues on Windows (https://www.youtube.com/watch?v=E6hjKrruiIY), other platforms are even easier.

AMSjobs can submit and monitor jobs for you on different queues, both locally and remotely. You can use the GUI on your desktop or laptop for creating, submitting and analyzing your jobs, while running the jobs on remote compute clusters. In that case you should establish and ssh-agent connection, to enable remote job managing by the GUI. If you just run local jobs without a queuing system, skip this section.

To set up a new batch queue, select **Queue → New… →** and choose an example queuing system to starting with (LSF, PBS, or SGE). Modify the input files such that they correspond to your queue configuration:

- change the **Name** of the queue
- set the **Remote host** to the hostname of your cluster
- set **Remote user** to your username on that cluster
- change the **Run command** in accordance with your typical queue set up. The **$options** corresponds to the input box that can be modified in AMSjobs before submitting, e.g. the time or number of nodes
- you can set what $options defaults to in the **Default Options** field
- change the **Kill**, **Job status**, **System status commands**, if necessary
- you may define a **Prolog** or **Epilog** command

5.7.2 Managing remote jobs

To manage remote jobs, you need automatic authentication with an ssh-agent via an ssh key pair. This is most easily set up on MacOS or Linux, but is a bit more involved for Windows.

ssh connection on MacOS and Linux

If you don't have an ssh key pair already, you can generate one in a terminal: ssh-keygen -t rsa Put your pass-word protected public key on the compute cluster(s) in ~/.ssh/authorized_keys.

Next, you should establish a connection via an ssh-agent. On MacOS an ssh-agent runs by default, and with keychain you just have to type your password once when you make the first connection.

On various Linux installations ssh-agent is also running by default. If an ssh-agent daemon is not yet running in the terminal where you will run AMSjobs, start the daemon, e.g. by adding eval $(ssh-agent bash) to your .bashrc. You need to add your private key to the ssh-agent daemon: ssh-add.

AMSjobs will use SSH to connect to the remote systems, making a new connection for everything it does. This means that there will be many connections to the remove machine(s). If you are using OpenSSH (both on the local and the remote machine), you can instead use one ssh connection and keep it alive, reusing it for many things. This makes AMSjobs faster, and may avoid complaining system administrators of the remote machines.

To do this, set the environment variable SCM_SSH_MULTIPLEXING to yes (for example via the AMSprefs application).

ssh connection on Windows

PuTTY tools, automatically installed with ADF, can be used to set up an ssh-agent connection. Follow these steps, using the PuTTY programs located in the bin\Putty directory in your AMS installation directory.

- Run puttygen.exe, set the key length to at least 2048 bits and press the Generate button. This will generate an ssh key pair. Type a key passphrase and save the private key to a file. Add the corresponding public key to the ~/.ssh/authorized_keys file on the remote host.
- Run pageant.exe and add your private ssh key (right-click on the Pageant icon in the task bar) with the correct passphrase.
- Open AMSjobs menu Help|Command-line and run "ssh user@host uptime" using correct username and hostname. Type **yes** if prompted to store the key in cache. If you get prompted for a password or get authentication errors then something is not configured correctly. The remote job submission will not work until all problems are resolved.

Now you can close the prompt and start using AMSjobs to manage remote jobs. You may test your queue configuration: **Jobs → Generate Test Job** and assign it to your new queue before running the test job.

Centrally defined queues

A system administrator may also define a queue centrally, which may then be picked up by any user automatically via Dynamic queues. Consult the GUI manual for information on how to set these up in AMSjobs.

ffmpeg

The AMSmovie GUI module can make MPEG videos of the system as you see it in the AMSmovie window. This can be an MD trajectory or vibrational modes, or the course of geometry optimization. For this to work, you need to install the free ffmpeg program from the FFmpeg website (https://ffmpeg.mplayerhq.hu/) and make sure it can be found in the path. The simplest way to do this is to copy the ffmpeg executable to $AMSBIN.

CHAPTER

SIX

PACKAGE MANAGER: INSTALLING OPTIONAL COMPONENTS

This guide describes how to install optional components of the Amsterdam Modeling Suite. To reduce the initial download size, the Amsterdam Modeling Suite does not include all components by default. This includes features such as

- the Ligand Field DFT atomic database
- the COSMO-RS compound Database
- Quantum ESPRESSO
- Machine Learning potential engines

A full list of packages is available in the *FAQ section below* (page 34).

If the appropriate license is present, these optional components can be installed through the GUI or the command line. The easiest way requires an active internet connection, but it is also possible to use a local copy of our repository downloaded ahead of time. These instructions assume that you have already successfully installed AMS.

Note: AMSpackages behavior changed significantly in AMS2026. This page describes the package manager for AMS2026 and later. If you are using AMS2025 or an older release, see the documentation for previous AMS versions (https://www.scm.com/downloads/documentation-previous-versions/).

6.1 What's new

New in version AMS2026.101:

- The default behavior of AMSpackages now installs packages for all users.
- The default location for shared packages is now inside `AMSHOME` on all platforms.
- A `--single-user`/`-u` flag was added to fall back to the old per-user behavior.

The general installation procedure should be simpler, especially for shared installations. This page now covers the recommended way to install optional AMS components. Use it for standard installs (GUI/CLI), and first-line troubleshooting. For advanced configuration and full command reference, see *Appendix D: Advanced package manager usage* (page 81).

6.2 Breakdown

There are two different install modes:

- **Admin mode (recommended):** Use this when you can write to AMSHOME.
- **Single-user mode:** Use this when you are not allowed to write to AMSHOME.

AMSpackages automatically selects admin mode if you are allowed to write to AMSHOME. Otherwise, it falls back to single-user mode.

6.3 Recommended Installation

For most users and shared installations, admin mode is recommended: packages are installed in AMSHOME, keeping one managed package set for all users. If AMSHOME is not writable, the package manager defaults to single-user mode.

6.3.1 Pre-checks

Before installing, confirm:

- Your AMS license includes the package(s) you want.
- You have write permissions to AMSHOME if you use admin mode.
- The machine can access the package repository (for online installs).

6.3.2 GUI users

In AMSjobs, open SCM -> Packages. The package manager defaults to admin mode if you have permission to modify AMSHOME. Otherwise, it falls back to single-user mode. You can see the active mode in the AMSpackages window. You can use the GUI to search for the packages you need, select and install them. It will also show you which packages are already installed and if there are any updates.

You can find more GUI details in the AMSpackages GUI section of the documentation.

6.3.3 Command-line users

When installing packages:

```
"${AMSBIN}/amspackages" install lfdft
```

This installs into AMSHOME by default, unless that location is not writable.

Typical CLI flow is: view available packages, install, and verify:

```
"${AMSBIN}/amspackages" list
"${AMSBIN}/amspackages" install lfdft
"${AMSBIN}/amspackages" check lfdft && echo "lfdft was installed!"
```

6.3.4 Single-user mode

Single-user mode is only recommended when you are not allowed to write to `AMSHOME`. It will be enabled by default if this is the case. If you want to force single-user mode, you can use the `-u` flag.

```
"${AMSBIN}/amspackages" -u install lfdft
```

6.3.5 Copy-paste commands

Use these commands as a quick start:

```
# Install one package in default mode
"${AMSBIN}/amspackages" install lfdft

# Install multiple packages in default mode
"${AMSBIN}/amspackages" install lfdft qe

# Check whether a package is installed
"${AMSBIN}/amspackages" check lfdft

# Show install location for a package
"${AMSBIN}/amspackages" loc lfdft
```

6.4 Offline Usage Guide

Offline or mirrored package installs are useful, for example:

- when the target system cannot access the internet directly
- when computers are behind a firewall
- when you want to reuse one prepared local repository for repeated installations

Offline installs are detected automatically when the `AMSHOME` installation contains a folder named `repository` with `AMS2026.1.yml` at its top level.

AMSpackages detects and uses it without extra flags in that common case.

Local mirrors can be prepared with `amsrepomirror`, a separate tool for mirroring the SCM package repository. If you do not use that tool, a mirror can also be prepared manually.

To verify this on the command line:

```
"${AMSBIN}/amspackages" list
```

In the output header, verify that `Repository:` points to your local mirror location.

For instructions on creating or updating a local mirror, see *Appendix E: Mirroring the SCM package repository* (page 91). For advanced repository configuration in AMSpackages itself, see *Appendix D: Advanced package manager usage* (page 81).

6.5 FAQ and Troubleshooting

6.5.1 What packages are available?

At the time of writing, the following packages are available:

Table 6.1: Available AMSpackages

Package name	Package ID	Available version
ADFCRS-2018 Database	adfcrs	2018: build 5
AIMNet2 Environment (CPU Only)	aimnet2-cpu	1.0.0: build 224
AIMNet2 Environment (CUDA 12.8)	aimnet2-cu128	1.0.0: build 224
Amsterdam Modeling PySuite Environment	amsterdam-modeling-pysuite	1.0: build 224
FAIRChem Environment (CPU Only)	fairchem-cpu	1.0.0: build 224
FAIRChem Environment (CUDA 12.8)	fairchem-cu128	1.0.0: build 224
Infretis Environment	infretis	1.0.0: build 224
LFDFT atomic database	lfdft	1.0: build 0
M3GNet Environment	m3gnet	1.0.0: build 224
MACE Environment (CPU Only)	mace-cpu	1.0.0: build 224
MACE Environment (CUDA 12.8)	mace-cu128	1.0.0: build 224
Subgraph Sigma Profile Estimation (SG1) Database	molsg_sg1db	1.0: build 3
NequIP Environment (CPU Only)	nequip-cpu	1.0.0: build 224
NequIP Environment (CUDA 12.8)	nequip-cu128	1.0.0: build 224
OLED material database	oledmatdb	2023.1: build 2
Quantum ESPRESSO (AMSPIPE)	qe	7.1: build 206
TorchANI Environment (CPU Only)	torchani-cpu	1.0.0: build 224
TorchANI Environment (CUDA 12.8)	torchani-cu128	1.0.0: build 224
Zacros-post	zacros_post	1.02: build 2

6.5.2 What are the most common pitfalls?

The most common installation problems are:

- Assuming a default install is always shared/admin mode. If `AMSHOME` is not writable, AMSpackages falls back to single-user mode.
- Trying to install a package that is not included in your license.
- Trying an online install from a machine behind a firewall or without internet access. In that case, prepare a local mirror and follow the offline setup described above.

6.5.3 Install failed / permission denied

If installation fails with a permission error, verify whether you are installing in admin mode. Admin mode requires write permissions to `AMSHOME`. If you cannot get write access, rerun the installation in single-user mode with `-u` or `--single-user`.

6.5.4 Package not found

If a package does not appear in `"${AMSBIN}/amspackages" list`, check that:

- The package is included in your license.
- You are using the correct AMS version and repository.
- You are not accidentally using an outdated local offline mirror.

6.5.5 Offline repo not detected

Make sure the folder name is exactly `repository` and that `AMS2026.1.yml` is directly inside that folder. The `repository` folder must be inside `AMSHOME` for automatic detection. Then rerun:

```
"${AMSBIN}/amspackages" list
```

and verify that the `Repository:` line points to your local mirror. Alternatively, you can export the `SCM_AMSPKGS_REPO` variable to point to the `AMS2026.1.yml` file in an alternative location.

6.5.6 The package manager exits with "Unhandled exception..."

The package manager has encountered an unexpected error. To see more information, the command-line version of AMSpackages has a `--verbose` option. Using this flag twice yields extra debug output (`-vv`), which may include more information about the error.

Some errors may disappear after running `amspackages clean` on the command line. If errors persist, please contact SCM support and include logs.

Note: If you face problems with the package manager, please include the log file in your support ticket. The log file is called `amspackages.log` and is located under:

- Linux: `$HOME/.scm/packages/amspackages.log`
- MacOS: `$HOME/Library/Application Support/SCM/packages/amspackages.log`
- Windows: `%LOCALAPPDATA%/SCM/packages/amspackages.log`

Alternatively, run the command below and send `debug_info.txt`: `$AMSBIN/amspackages debug > debug_info.txt`

6.5.7 Where are packages installed?

By default, AMSpackages will attempt to install packages inside of `AMSHOME`.

In admin mode, packages are installed in the shared installation inside `AMSHOME` by default.

In single-user mode, AMSpackages uses the following default locations:

- Linux: `$HOME/.scm/packages`
- MacOS: `$HOME/Library/Application Support/SCM/packages`
- Windows: `%LOCALAPPDATA%/SCM/packages`

Every version of AMS has its own dedicated folder inside the `packages` directory, to allow different installations to appear side-by-side. Packages themselves are located in uniquely named subdirectories. If you need the location of a specific package, use:

```
"${AMSBIN}/amspackages" loc lfdft
```

The single-user default location can be changed using `SCM_AMSPKGS_USERDIR`. The shared admin location can be changed using `SCM_AMSPKGS_SHAREDDIR`.

6.6 Need more control?

For advanced usage, see *Appendix D: Advanced package manager usage* (page 81), including:

- advanced CLI commands (install/update/remove and full options)
- persistent configuration and environment variables
- advanced offline mirror creation methods
- admin Python package management details

CHAPTER

SEVEN

ADDITIONAL INFORMATION AND KNOWN ISSUES

7.1 Windows Subsystem for Linux (WSL) and Docker

AMS does **NOT** run in parallel under WSL1, WSL2, or Docker on Windows, because the IntelMPI library does not support these configurations.

7.2 Shared memory and batch systems (SLURM)

When discussing memory usage by a parallel job one should distinguish the private and shared memory. The private memory is, well, private to each process and to get the total job's memory usage one should add up the sizes of private memory segments from all processes. The shared memory is used by several processes on a shared-memory node, so for the job's total one should add each shared memory segment only once. However most batch systems ignore the fact that shared memory segments are shared and instead of adding it to the total on the per-node basis they add it per-process. This leads to the batch system greatly overestimating the memory consumption for jobs that use a lot of shared memory, for example, ADF calculations with hybrid functionals. This may lead to such jobs being killed and/or the users being over-charged or forced to use the more expensive hugemem queues. This would be totally unnecessary if the job accounting was properly configured. Luckily, SLURM has the JobAcctGatherParams=UsePss (https://slurm.schedmd.com/slurm.conf.html) option (off by default) that is supposed to take care of the private vs shared memory difference and ensure the correct accounting of ADF jobs. You can check the status of JobAcctGatherParams on your SLURM system with the following command:

```
scontrol show config | grep JobAcctGatherParams
```

7.3 GPFS file system

Starting with AMS2019, the KF sub-system (used for handling binary files such as ADF's TAPE* files) has been rewritten to use memory-mapped files. The mmap() system call implementation is file-system dependent and, unfortunately, it is not equally efficient in different file systems. The memory-mapped files implementation in GPFS is extremely inefficient. Therefore the users should avoid using a GPFS for scratch files.

7.4 Running MPI jobs

Most programs in the Amsterdam Modeling Suite (AMS) are MPI-enabled, and they are set up to run in parallel automatically, both from the GUI and the command line. For example, to run a parallel AMS calculation from the command line, simply do:

```
$AMSBIN/ams < myinputfile
```

A common mistake by people who have worked with MPI programs before is to "mpirun" the programs manually, which results in a parallel mpirun call and usually a lot of errors. All MPI programs in the AMS suite are executed via *$AMSBIN/start* to set up the environment and call mpirun for you. If you need to modify the *mpirun* command, please edit the *$AMSBIN/start* script.

7.4.1 Technical explanation

In the example above, AMS is started in parallel, which involves a couple of files:

- $AMSBIN/ams: a symbolic link to the $AMSBIN/start script
- $AMSBIN/start: this script sources $AMSBIN/setenv.sh to set up most of the environment, determines the name of the used symlink, and then starts the related binary (ams.exe in this case) via mpirun. On Linux clusters it also attempts to detect the scheduler to launch the MPI program with the allocated resources.
- $AMSBIN/setenv.sh: when sourced, this script sets up the correct environment for locating libraries and runtimes such as MKL and IntelMPI
- myinputfile: A valid input file. Some examples can be found in $AMSHOME/examples, and of course the input is also documented per program in the manual.

7.5 More on running MPI jobs

MPI (Message Passing Interface) is a standard describing how to pass messages between programs running on the same or different machines.

MPI is a formal standard and it is actively supported by all major vendors. Some vendors have highly-optimized MPI libraries available on their systems. There are also a couple of open-source implementations of the MPI standard, such as MPICH and OpenMPI. There are also numerous commercial MPI implementations that support a wide range of systems and interconnects, for example, HP-MPI (formerly known as Platform-MPI) and IntelMPI.

Support for a particular MPI implementation in ADF can be considered at three levels: the source code, the configure script, and pre-compiled binaries. At each level different MPI implementations may be supported.

The ADF source code is not implementation-specific and thus theoretically it supports any MPI library. Many popular MPI implementations are supported at the level of the configure script, but depending on your local setup you may need to make some modifications in the buildinfo file after running configure. For example on 64-bit Linux IntelMPI and OpenMPI should work directly, but using other MPI flavors will most likely require manual changes to the *buildinfo* file correct the include and linker paths to the MPI libraries of your system. The configure script will also try to generate an appropriate $AMSBIN/start script, but this might also need modification when using different MPI libraries. In general it is best to use the same MPI version used by SCM for the precompiled binaries.

When choosing an MPI implementation for pre-compiled binaries, SCM considers many factors including (but not limited to) the re-distribution policy, performance, and built-in support for modern interconnects. IntelMPI is currently the standard MPI implementation supported by SCM because it has the most favorable combination of these factors at this moment. For platforms where IntelMPI is supported its runtime is distributed with ADF (Windows, Linux). OpenMPI

builds are also available for linux, but should only be used in case of problems with IntelMPI. A different MPI implementation will be standard on a platform where IntelMPI is not available. It may or may not be distributed with ADF. For example, SGI MPT is standard on SGI machines and OpenMPI is standard on Mac OS X platforms, but only the latter is distributed together with ADF.

When pre-compiled binaries do not work on your computer(s) due to incompatibility of the standard MPI library with your soft- and/or hardware, the SCM staff will be glad to assist you in compiling ADF with the MPI implementation supported on your machine(s).

If you are going to use an MPI version of the ADF package, and it is not IntelMPI or OpenMPI, you will need to determine if the corresponding MPI run-time environment is already installed on your machine. If not, you will need to install it separately from ADF. As it has been already mentioned, IntelMPI and OpenMPI are bundled with the corresponding version of ADF so you don't need to worry about installing them separately.

Running with MPI on more than one node

When running on more than one machine (for example on a cluster **without** a batch system) you need to specify a list of hosts on which mpirun needs to spawn processes. In principle, this is implementation-specific and may be not required if the MPI is tightly integrated with your operating and/or batch system. For example for MPICH1 you can do this by preparing a file containing hostnames of the nodes (one per line) you will use in your parallel job. Then you set the SCM_MACHINEFILE environment variable pointing to the file.

When you submit a parallel job to a batch system the job scheduler usually provides a list of nodes allocated to the job. The $AMSBIN/start shell script has some logic to extract this information from the batch system and pass it to the MPI's launcher command (typically mpirun). In some cases, depending on your cluster configuration, this logic may fail. If this happens, you should examine the $AMSBIN/start file and edit the relevant portion of it. For example, you may need to add commands that process the batch system-provided nodelist or change mpirun's command-line options or even replace the mpirun command altogether.

7.6 IntelMPI and core-binding

IntelMPI by default uses core binding for the spawned processes (also known as process pinning). This can be disabled by setting the **I_MPI_PIN** environment variable to "off".

7.7 IntelMPI and SLURM

To get IntelMPI work under SLURM one needs to set the I_MPI_PMI_LIBRARY environment variable to the correct path for the libpmi.so library from SLURM. If your SLURM system is configured to use PMI2, then it could also be sufficient to ignore the I_MPI_PMI_LIBRARY.

Depending on your SLURM version and configuration, it might be necessary to use the IntelMPI 2021 runtime (export SCM_USE_IMPI_2021=1), or use a local IntelMPI runtime from the modules system (export SCM_USE_LOCAL_IMPI=1, and make sure to load the local IntelMPI module).

If this does not help, you might need to check if the SLURM configuration has a default MPI selected. To check this, execute **scontrol show config | grep Mpi** and look for MpiDefault. This default can be overridden with the **srun –mpi=** option, which can be set using the SCM_MPIRUN_OPTIONS environment variable. See the SLURM srun manual page for further details.

When running under SLURM, if you see messages such as `libmpifort.so.12: cannot open shared object file: No such file or directory`, make sure that **SLURM_EXPORT_ENV** is not set to **None**, or use `export SCM_MPIRUN_OPTIONS="--export=ALL"`.

7.8 IntelMPI and SGE

To get IntelMPI working with Sun Grid Engine, one has to define a parallel environment. How this can be done is described on the intel website (https://software.intel.com/en-us/articles/integrating-intel-mpi-sge). It is important for modern versions of IntelMPI (as used in AMS2021) and newer to make sure to set "job_is_first_task FALSE" in the parallel environment, otherwise jobs will fail to start.

7.9 IntelMPI and ABI compatibility

IntelMPI v5.0 or newer is ABI (Application Binary Interface) compatible with Cray MPT v7.0.0 or newer and MPICH v3.1 and newer. This means that binaries compiled with one of these libraries can use the other ones during run-time without problems. Our IntelMPI binaries should work out-of-the-box on Cray machines using the ABI compatibility, and can also be used in combination with MPICH 3.2.

To run ADF with MPICH instead of IntelMPI, simply **export SCM_USE_LOCAL_IMPI=true**, and make sure the MPICH mpirun command is available in your PATH variable. Core binding (process pinning) is disabled by default for MPICH, to enable this add "-bind-to core" to the mpirun commands in the $AMSBIN/start file.

7.10 Multi-node issues

A common reason for multi-node jobs failing where single-node jobs work, is a missing scratch folder on the secondary nodes. Especially SLURM systems are known to sometimes only create the TMPDIR folder on the first node of the job. To solve this, make sure that the SCM_TMPDIR exists before starting the AMS/ADF calculation. For example, under SLURM you could try "srun –ntasks-per-node=1 mkdir -p $SCM_TMPDIR" from your submit script, or even incorporate it into the $AMSBIN/start script.

7.11 OpenMPI on Linux

The OpenMPI 4.1.1 binaries supplied with AMS2025 should work on desktop, laptop and workstation machines out of the box (single-node usage). On cluster environments it might be necessary to compile an OpenMPI 4.1 library with support for the cluster queueing system and/or the infiniband solution. Make sure to **export SCM_USE_LOCAL_OMPI=true** before starting programs to enable your local OpenMPI version instead of the one shipped with ADF. Core binding (process pinning) is enabled by default for OpenMPI, to disable this, export SCM_MPIOPTIONS="–bind-to none" into your environment before starting AMS.

7.12 Corrupted License File

You may find that, after having installed the license file, the program still does not run and prints a message "LICENSE CORRUPT". There are a few possible causes. To explain how this error may come about, and how you overcome it, a few words on license files.

Each license file consists of pairs of lines. The first of each pair is text that states in a human-readable format a couple of typical aspects: A 'feature' that you are allowed to use (for instance 'ADF'), the expiration date, a (maximum) release (version) number of the software and so on. The second line contains the same information in encrypted format: a long string of characters that appear to make little sense. The program reads the license file and checks, with its internal encrypting formulas, that the two lines match. If not, it stops and prints a "LICENSE CORRUPT" message.

So, there are two common reasons why this may happen:

You can use the **fixlic** utility to try to fix this automatically. Please be aware that the **fixlic** utility will try to fix the file pointed to by the $SCMLICENSE environment variable and replace it with the fixed copy. Thus, you need to make a backup of your license file first and you need to have write permissions for it.

```
cp $SCMLICENSE $SCMLICENSE.backup
$AMSBIN/fixlic
```

7.13 Windows: running jobs from the command line

In order to run ADF or any other program from the package without the GUI, navigate to the ADF installation directory and double click the **adf_command_file.bat** file. It will start a Windows command interpreter and set up the environment specific for that installation of ADF. Once it has started, cd to your jobs directory by entering the following commands at the prompt:

```
C:
cd \ADF_DATA
```

Then, run your job as follows (assuming the job is called h2o):

```
sh h2o.job
```

You can also prepare a job from a .ams file and run it using only two commands:

```
sh amsprep -t h2o.ams -j h2o > h2o.job
sh h2o.job
```

Please note that you do need to use *sh* in the commands above because both h2o.job and amsprep are shell scripts and, thus, they must be interpreted by a shell.

If you are comfortable with a UNIX shell environment, you can also start a bash shell and enjoy a basic msys2 LINUX environment:

```
bash
```

7.14 Windows: Installing a license from the command line

To install a license from the AMS command line, you can run the following autolicense command.

```
sh autolicense nogui -u username -p password -m mailaddress
```

System administrators may be interested in running this directly from a .bat file, in order to automate installation on several computers. The windows installation of AMS comes with a `licensetool.bat` script inside the top `%AMSHOME%` directory (e.g. `C:\AMS2022.101\licensetool.bat`), which is used for running license requests from within AMS. We don't recommend changing this file itself, but you can make a copy of it for other purposes. If you wish to update the license non-interactively, you can replace the autolicense command in your copy with

```
%DIR%\msys\usr\bin\sh.exe  %DIR%\bin\autolicense nogui -u username -p password -m␣
↪mailaddress
```

while filling in the correct username, password, and mail address for your license account. This will automatically install a license into the `%AMSHOME%` directory. The licensetool.bat script assumes that it itself is located in `%AMSHOME%` by default. The .bat file can also be run from elsewhere by using a single command-line argument.

```
customlicensetool.bat C:\PATH_TO\AMS2022.101
```

This directory has to be the AMSHOME folder, containing all of the AMS files, including the bin and msys folders.

CHAPTER

EIGHT

USING THE GUI ON A REMOTE MACHINE

If you encounter problems launching the GUI on a remote machine via x forwarding over SSH, usually the solution is to use our OpenGL Software rendered GUI instead. Set the SCM_OPENGL_SOFTWARE environment variable before launching the GUI to activate software rendering for the GUI:

```
export SCM_OPENGL_SOFTWARE=1
amsjobs &
```

If this does not help you, please read the rest of this document.

8.1 More detailed information

Running the ADF GUI (amsinput) on a remote machine can be accomplished in two ways. The most simple way is using x forwarding over SSH. The second way is using a VNC setup like Cendios ThinLinc (see *Using ThinLinc for Remote GUI* (page 50)), which needs to be set up on the remote machine but offers better performance, especially for tasks that are heavy in graphics and interactivity. *ADF2017 and VTK7* (page 49).

8.2 X11 over SSH

When connecting to a remote system (let's call it RemoteBox) from your local machine (LocalBox) over SSH, there is the option to enable X forwarding:

```
ssh -X -Y user@RemoteBox
```

This allows you to run X11 GUI programs on RemoteBox while the window shows on LocalBox. You can try some simple X11 programs now:

```
xterm
xclock
```

If you got a terminal and a clock popping up in separate windows, you have working X11 forwarding over SSH.

If you got any errors and no window, most likely something fundamental is broken. Check if the RemoteBox allows X11 Forwarding (/etc/ssh/sshd_config should contain "X11Forwarding yes"), and try both the -X and -Y flag on the ssh command. Another thing to check is "xhost", which might also get in the way.

sidenote on ssh -X and -Y: There are two ways of enabling X forwarding with SSH, -X and -Y. The default -X flag enables X11 forwarding, but with extended security restrictions to protect LocalBox from malicious behavior of people on the RemoteBox. The -Y flag enables trusted X11 forwarding, which does not enable the additional security extensions. The security extensions are there for good reason, but they can break so many things that some distros (Debian for example)

disable them by default, even if you use -X and not -Y. Which mode you decide to use is up to you of course, but if something doesn't work always try using -X -Y (or -XY) before asking for help.

Attention: If you did not succeed in getting an xterm or xclock window to show, then you MUST solve that problem before trying to use 3D graphics. The rest of this text assumes you have a working X11 setup.

8.3 X11 with OpenGL over SSH (3D graphics)

X11 over SSH can also support OpenGL 3D graphics, using the GLX extensions. To test this run:

```
glxgears
```

It should pop up a window with 3 gears, which may or may not be spinning. To get more info about the 3D driver stack, run:

```
glxinfo |  grep -E " version| string| rendering|display"
```

If the above two commands produce errors (For example: `Error: couldn't find RGB GLX visual or fbconfig` or `X Error of failed request:`) something fundamental is broken with the 3D setup on RemoteBox or LocalBox. You should check the 3D driver stack on **both** machines, and pay special attention to the libGL.so and libGLX*.so libraries. Make sure you can run `glxgears` and `glxinfo` on LocalBox before investigating the remote machine.

8.3.1 Intel Graphics (mesa)

If LocalBox has intel graphics, you might run into problems if RemoteBox uses proprietary hardware & drivers. First check which libGL.so is used on RemoteBox by logging in via SSH and running:

```
ldd `which glxinfo`
```

The output will contain a line similar to:

```
libGL.so.1 => /usr/lib/x86_64-linux-gnu/libGL.so.1 (0x00007f76cdcaf000)
```

/usr/lib/x86_64-linux-gnu/libGL.so.1 is most likely a symlink, so use `ls -l` to find its true destination:

```
ls -l /usr/lib/x86_64-linux-gnu/libGL.so.1
```

if this points to a proprietary graphics library (for example: `lrwxrwxrwx 1 root root 30 aug 22 13:24 /usr/lib/x86_64-linux-gnu/libGL.so.1 -> /usr/lib/nvidia-361/libGL.so.1`), you can try pre-loading the mesa version of the library on the remote machine. Try to locate the mesa libGL.so:

```
find /usr -iname "*libGL.so*" -exec ls -l {} \;
```

If we are lucky we spot the mesa libGL.so in the output. It may look something like this:

```
/usr/lib/x86_64-linux-gnu/mesa/libGL.so
```

If you found the mesa libGL.so, try to put it in LD_PRELOAD:

```
export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/mesa/libGL.so
glxinfo
```

If the system was already using the mesa libGL.so, you can try to use indirect rendering by setting the following environment variable:

```
export LIBGL_ALWAYS_INDIRECT=1
```

Other useful environment variables can be:

```
export LIBGL_DEBUG=verbose
export LIBGL_ALWAYS_SOFTWARE=1
```

8.3.2 NVidia Graphics

There are probably two libGL implementations on LocalBox if it uses the NVidia proprietary drivers (or 4 if you also have 32bit libraries installed): the opensource "mesa" library (`libGL.so.1.2.0`, most likely in a "mesa" subdirectory) and the proprietary NVidia library that comes with the NVidia drivers. Run the following command to find the libGL libraries on your system:

```
find /usr -iname "*libGL.so*" -exec ls -l {} \;
```

Make sure that the `libGL.so` and `libGL.so.1` symlinks in the generic folders (/usr/lib/, /usr/lib64, /usr/lib/x86_64_linux_gnu) eventually point towards the proprietary library. You will need to have root permissions to change the symlinks.

Warning: Never run any commands as root (or with sudo) to change your system setup if you do not understand them. It is not hard to break a Linux installation when making mistakes as root, so make backups before you change something and double-check what you type when using sudo or the root account!

Another important library when running OpenGL over SSH with NVidia hardware is `libGLX.so`. Locate it on your system with the following command:

```
find /usr -iname "*libGLX*.so*" -exec ls -l {} \;
```

Make sure that there are symlinks to the proprietary library in a generic location (/usr/lib on ubuntu).

You can check if the correct `libGL.so` is being used by checking the dynamic library dependencies of glxinfo:

```
ldd `which glxinfo`
```

The reported `libGL.so` dependency is most likely a symlink, so use `ls -l` on it to find out where it points to.

libGL.so examples

A correct setup on CentOS 6 with NVidia drivers for example should look something like this:

```
# first 3 lines are the NVidia lib
# second set of 3 lines are the mesa lib
# last two lines are the generic symlinks that point towards the NVidia lib
-rwxr-xr-x 1 root root 1220472 apr  6 02:51 /usr/lib64/nvidia/libGL.so.352.93
lrwxrwxrwx 1 root root 15 aug 19 13:33 /usr/lib64/nvidia/libGL.so.1 -> libGL.so.352.93
lrwxrwxrwx 1 root root 15 aug 19 13:33 /usr/lib64/nvidia/libGL.so -> libGL.so.352.93
-rwxr-xr-x 1 root root 561640 mei 11 06:38 /usr/lib64/mesa/libGL.so.1.2.0
lrwxrwxrwx 1 root root 14 jun  6 13:40 /usr/lib64/mesa/libGL.so.1 -> libGL.so.1.2.0
```

(continues on next page)

(continued from previous page)

```
lrwxrwxrwx 1 root root 14 jun  6 13:40 /usr/lib64/mesa/libGL.so -> libGL.so.1.2.0
lrwxrwxrwx 1 root root 15 aug 19 13:48 /usr/lib64/libGL.so -> nvidia/libGL.so
lrwxrwxrwx 1 root root 17 aug 19 13:48 /usr/lib64/libGL.so.1 -> nvidia/libGL.so.1
```

Another example of an 64bit ubuntu installation with NVidia drivers and 32bit libraries available:

```
-rw-r--r-- 1 root root 439972 jul 18 05:47 /usr/lib32/nvidia-361/libGL.so.1.0.0 #␣
↪32bit nvidia lib
lrwxrwxrwx 1 root root 10 aug  3 06:14 /usr/lib32/nvidia-361/libGL.so -> libGL.so.1
lrwxrwxrwx 1 root root 14 aug  3 06:14 /usr/lib32/nvidia-361/libGL.so.1 -> libGL.so.1.
↪0.0
lrwxrwxrwx 1 root root 10 jun 13  2013 /usr/lib32/libGL.so -> libGL.so.1 #generic␣
↪32bit symlink, points to the next line
lrwxrwxrwx 1 root root 21 aug 22 13:23 /usr/lib32/libGL.so.1 -> nvidia-361/libGL.so.1
↪# generic 32bit symlink, points to nvidia
-rw-r--r-- 1 root root 448200 jul 22 09:53 /usr/lib/i386-linux-gnu/mesa/libGL.so.1.2.
↪0 # 32bit mesa lib
lrwxrwxrwx 1 root root 14 jul 22 09:53 /usr/lib/i386-linux-gnu/mesa/libGL.so.1 ->␣
↪libGL.so.1.2.0
-rw-r--r-- 1 root root 579760 jul 18 05:50 /usr/lib/nvidia-361/libGL.so.1.0.0 # 64bit␣
↪nvidia lib
lrwxrwxrwx 1 root root 10 aug  3 06:14 /usr/lib/nvidia-361/libGL.so -> libGL.so.1
lrwxrwxrwx 1 root root 14 aug  3 06:14 /usr/lib/nvidia-361/libGL.so.1 -> libGL.so.1.0.
↪0
-rw-r--r-- 1 root root 459392 jul 22 09:52 /usr/lib/x86_64-linux-gnu/mesa/libGL.so.1.
↪2.0 # 64 bit mesa lib
lrwxrwxrwx 1 root root 14 jul 22 09:52 /usr/lib/x86_64-linux-gnu/mesa/libGL.so ->␣
↪libGL.so.1.2.0
lrwxrwxrwx 1 root root 14 jul 22 09:52 /usr/lib/x86_64-linux-gnu/mesa/libGL.so.1 ->␣
↪libGL.so.1.2.0
lrwxrwxrwx 1 root root 10 aug 22 13:25 /usr/lib/x86_64-linux-gnu/libGL.so -> libGL.so.
↪1 # generic 64bit symlink, points to next line
lrwxrwxrwx 1 root root 30 aug 22 13:24 /usr/lib/x86_64-linux-gnu/libGL.so.1 -> /usr/
↪lib/nvidia-361/libGL.so.1 # generic 64bit symlink, points to nvidia
```

8.3.3 AMD Graphics

AMD GPUs are reasonably well supported by the latest versions of mesa (ubuntu 16.04), and installing the proprietary Catalyst driver is not always a good idea. If you have an older distro (CentOS 6) you can install the fglrx Catalyst drivers. As a rule of thumb run the following command first:

```
glxinfo |  grep -E " version| string| rendering|display"
```

If this reports an OpenGL core profile version string of 4.x, do not install Catalyst. If the OpenGL core profile version string says 3.0, then check on google if fglrx is safe to install for your distribution.

libGL.so examples

An example of a CentOS 6 setup with AMD drivers should look a bit like this:

```
# first 4 lines are the 32bit AMD lib and symlinks
# next 4 lines are the 64bit AMD lib and symlinks
# last line is the renamed 64bit mesa library (renamed by the AMD driver installation)
-rwxr-xr-x. 1 root root 612220 Dec 18  2015 /usr/lib/fglrx/fglrx-libGL.so.1.2
lrwxrwxrwx. 1 root root 19 Aug 30 08:53 /usr/lib/libGL.so -> /usr/lib/libGL.so.1
lrwxrwxrwx. 1 root root 21 Aug 30 08:53 /usr/lib/libGL.so.1 -> /usr/lib/libGL.so.1.2
lrwxrwxrwx. 1 root root 33 Aug 30 08:53 /usr/lib/libGL.so.1.2 -> /usr/lib/fglrx/fglrx-
↪libGL.so.1.2
-rwxr-xr-x. 1 root root 921928 Dec 18  2015 /usr/lib64/fglrx/fglrx-libGL.so.1.2
lrwxrwxrwx. 1 root root 21 Aug 30 08:53 /usr/lib64/libGL.so -> /usr/lib64/libGL.so.1
lrwxrwxrwx. 1 root root 23 Aug 30 08:53 /usr/lib64/libGL.so.1 -> /usr/lib64/libGL.so.
↪1.2
lrwxrwxrwx. 1 root root 35 Aug 30 08:53 /usr/lib64/libGL.so.1.2 -> /usr/lib64/fglrx/
↪fglrx-libGL.so.1.2
-rwxr-xr-x. 1 root root 561640 May 11 06:38 /usr/lib64/FGL.renamed.libGL.so.1.2.0
```

8.3.4 OpenGL direct or indirect rendering

OpenGL with X11 can run in two different modes: direct rendering and indirect rendering. The difference between them is that indirect rendering uses the GLX protocol to relay the OpenGL commands from the program to the hardware, which limits OpenGL capabilities to OpenGL 1.4.

When using OpenGL over X11 with SSH, quite often direct rendering is not available and you have to use indirect rendering. Unfortunately indirect rendering is not always secure, so it got disabled by default on many recent Linux Distros. If you are using the mesa `libGL.so`, you can run the following commands to test if indirect rendering is working on LocalBox:

```
glxinfo
export LIBGL_ALWAYS_INDIRECT=1
glxinfo
```

If glxinfo crashes with something like:

```
name of display: :0
X Error of failed request:  GLXBadContext
  Major opcode of failed request:  154 (GLX)
  Minor opcode of failed request:  6 (X_GLXIsDirect)
  Serial number of failed request:  34
  Current serial number in output stream:  33
```

after setting LIBGL_ALWAYS_INDIRECT=1, then you might need to enable indirect rendering on LocalBox.

8.3.5 enabling indirect rendering on Xorg 1.17 and newer

X 1.17 disables indirect rendering by default. Enabling it can be a bit tricky, because the xorg.conf flag is only available in 1.19 and newer, so you need to use the **+iglx** command line flag when starting the X server.

> **Warning:** Be careful when making changes as root! Running these commands is at your own risk, and you should not execute them if you do not understand what they do.

CentOS 6

The X server call is hardcoded in gdm, so we need to create a wrapper script around Xorg. Log in as root on a console (Ctrl+Alt+F1) or via SSH and do:

```
cd /usr/bin/
mv Xorg Xorg.bin
echo -e '#!/bin/sh\nexec /usr/bin/Xorg.bin +iglx "$@"' > Xorg
chmod +x Xorg
init 3 # this stops the X server
init 5 # this starts the X server again
```

Ubuntu 16.04

```
sudo nano /usr/share/lightdm/lightdm.conf.d/50-xserver-command.conf
```

change the last line from `xserver-command=X -core` to `xserver-command=X -core +iglx` restart the machine, or restart the X server with:

```
sudo service lightdm restart
```

OSX / MacOS

Mac OS X users who update to XQuartz 2.7.9 will discover that they cannot use GLX applications remotely any more. This includes the ADF-GUI. To solve, at this point in time: stick to the older version of XQuartz (2.7.8), or install the 2.17.10 beta version: https://www.xquartz.org/releases/XQuartz-2.7.10_beta2.html. After installing they should run this command to enable GLX:

```
defaults write org.macosforge.xquartz.X11 enable_iglx -bool true
```

8.3.6 OpenGL2+ with X11 over SSH

If you need OpenGL2+ features, there are two options: use direct rendering (this usually only works if both LocalBox and RemoteBox use a recent mesa libGL.so), or use `VirtualGL`. The VirtualGL tool (https://www.virtualgl.org/) intercepts the OpenGL calls, does the 3D rendering on RemoteBox and then transports the resulting image to LocalBox. For more details on how this is achieved see their documentation (https://cdn.rawgit.com/VirtualGL/virtualgl/2.5/doc/index.html).

Install VirtualGL on LocalBox and RemoteBox, and configure (run `vglserver_config` as root, see the VirtualGL documentation (https://cdn.rawgit.com/VirtualGL/virtualgl/2.5/doc/index.html#hd006%20to%20setup%20RemoteBox)) RemoteBox to operate as a VirtualGL server. RemoteBox should of course also have proper 3D hardware and drivers

(intel integrated graphics with recent mesa drivers, or a dedicated AMD/NVidia GPU), and be capable of indirect rendering (see above).

Once virtualGL is installed and set up on RemoteBox and installed on LocalBox, open a terminal window on LocalBox and connect to RemoteBox with:

```
vglconnect -s username@RemoteBox
```

This will start a VirtualGL client on LocalBox and set up two encrypted tunnels to RemoteBox. On RemoteBox you can now run OpenGL programs by starting them through `vglrun`:

```
vglrun glxinfo
vglrun glxgears
```

If you look at the output of the glxinfo command when running through vglrun, you will see that both the server and client glx vendor string changed into "VirtualGL", and the OpenGL renderer & version string should now say something about the hardware of RemoteBox.

8.3.7 ADF2017 and VTK7

ADF2017 uses VTK7 and newer OpenGL features to dramatically speed up the visualization of large systems. This comes with a higher requirement for the OpenGL version supported by the system: OpenGL 3.2. If your machine does not support this, you might get the following error message when starting the GUI:

```
Warning: In /home/builder/jenkins/workspace/trunk/label/centos6_impi_lxc/VTK/
↪Rendering/OpenGL2/vtkOpenGLRenderWindow.cxx, line 647
vtkXOpenGLRenderWindow (0x7b93c40): VTK is designed to work with OpenGL version 3.2␣
↪but it appears it has been given a context that does not support 3.2. VTK will run␣
↪in a compatibility mode designed to work with earlier versions of OpenGL but some␣
↪features may not work.
```

In such cases, a fallback mode is available that lowers the OpenGL requirement to version 1.4, but this fallback mode of course does not have the performance benefits of the newer OpenGL features. To enable the fallback mode, set the `SCM_OPENGL1_FALLBACK` environment variable to something non-zero:

```
export SCM_OPENGL1_FALLBACK=1
amsinput &
```

The OpenGL 3.2 requirement should not present any problems, unless your hardware or OS is really old. Known problematic cases are:

- CentOS 6 with intel graphics has OpenGL 2.1 (possible solutions: get an NVidia or AMD GPU with closed-source drivers, or use the fallback mode)
- OpenGL with X11 over SSH (possible solutions: use VirtualGL (see *OpenGL2+ with X11 over SSH* (page 48)) or the fallback mode)
- Remote Desktop on Windows: The 32bit Windows version of ADF2017.107+ has been made OpenGL 1.4 compatible to deal with older hardware and Remote Desktop problems. You can try to install the 32bit version of ADF2017 on your Windows machine if you have problems with starting the GUI on Windows.

8.3.8 AMS2019/AMS2018 and VTK7

AMS2019 uses the same VTK7 as AMS2018 and ADF2017, and thus also requires OpenGL 3.2 or newer. However, the OpenGL 1.4 fallback mode is now available on 64bit Windows and Linux, and should trigger automatically when it detects an OpenGL version on the system that is too old for the normal mode. The `SCM_OPENGL1_FALLBACK` environment variable can also still be used, for example when the detection is not working. Setting this on Windows can be done with the following steps: `Winkey+R`, `sysdm.cpl`, Advanced, Environment Variables… Here you can either add the `SCM_OPENGL1_FALLBACK` key with value 1 to the current user, or as admin for everyone on the system.

8.3.9 AMS2020/AMS2021/AMS2022 and VTK7

AMS2020/AMS2021/AMS2022 ships a software implementation of OpenGL on Linux and Windows. On Windows this is automatically enabled if the system has no support for OpenGL 3.2 or newer. The `SCM_OPENGL_SOFTWARE` environment variable can be set to use this on Linux, or force it on Windows. The `SCM_OPENGL1_FALLBACK` environment variable is no longer available.

8.4 Using ThinLinc for Remote GUI

Cendio ThinLinc (https://www.cendio.com/) is a secure remote access software for Linux built on open source technologies such as SSH and the VNC protocol. ThinLinc is a commercial product, but free to use for up to 10 users in an organization. It provides the ability to use a Linux desktop and Linux applications remotely and also comes with features like easy file transfer, sharing of the clipboard and others. Much of the performance gains that you may experience by using ThinLinc stem from using VNC instead of x forwarding as the underlying protocol and optimally configuring SSH to handle the VNC transport stream. This becomes especially evident if you are connecting to a remote system that is far away from your location.

If ThinLinc is not already installed on the remote system, you will first need to install the ThinLinc server on the remote machine. To install ThinLinc you need “root” or “sudo” privileges on the remote machine. Once the installation is complete, “root” or “sudo” are no longer needed. ThinLinc is free to use for up to 10 users in an organization and the server and client can be downloaded here (https://www.cendio.com/thinlinc/download).

Before installing the ThinLinc server, a desktop environment needs to be installed on the remote system, if it is not already installed. For example, you can follow this article (https://www.makeuseof.com/install-desktop-environment-gui-ubuntu-server/) to learn more about installing a desktop environment for Ubuntu systems. For performance reasons, we recommend installing the “ubuntu-mate-core” desktop environment or a similar lightweight environment. Once a desktop environment is installed, you can install the ThinLinc server, which will take care of all other dependencies and also configure the remote system. The ThinLinc server quickstart guide is available here (https://www.cendio.com/thinlinc/docs/install). Here is a YouTube video (https://youtu.be/lTdTZfjSRY4) showing a ThinLinc server and client installation.

After installing the ThinLinc server on the remote system, you can install the ThinLinc client on your local machine. ThinLinc clients are available for Windows, Mac OS and Linux, and can be downloaded from here (https://www.cendio.com/thinlinc/download). A client is not needed if you are using the web access feature of ThinLinc, which allows accessing the remote system through any modern web browser. To finish the configuration of the web access feature, please have a look at the server configuration.

Once the client is installed, you can connect to the remote machine using the same credentials that you would use to connect with SSH. ThinLinc supports connecting by User ID/Password or using a public/private key pair and also 2-factor authentication.

If you have trouble installing or using ThinLinc, please take a look at the community forum (https://community.thinlinc.com/). Cendio also offers commercial support for ThinLinc.

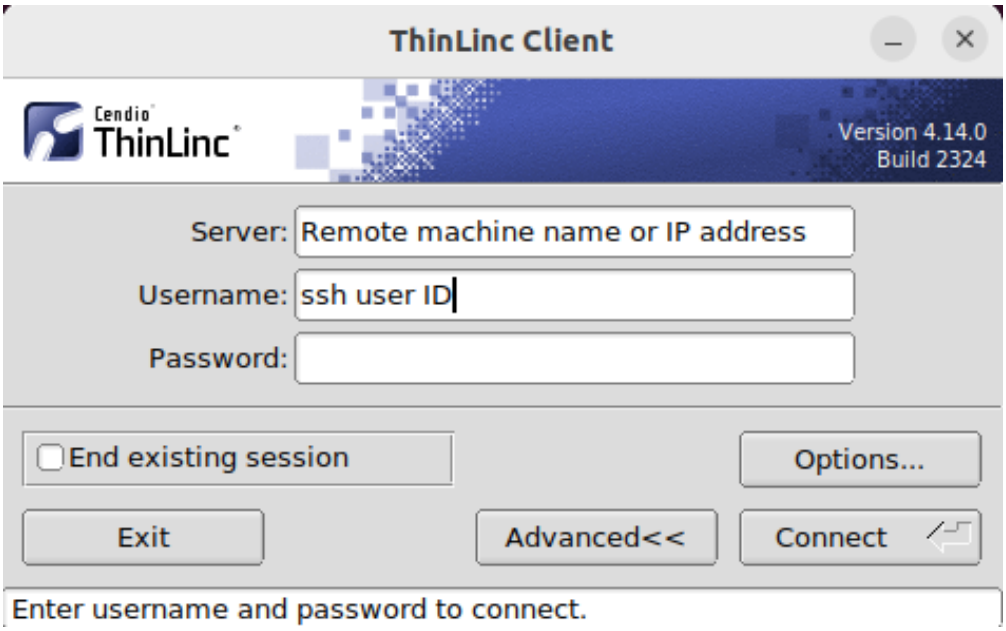


Fig. 8.1: ThinLinc Connect Dialog

8.5 Sources

The information on this page is a mashup of local knowledge, a lot of testing and reading on the web. The following pages contained some useful information:

- https://www.phoronix.com/scan.php?page=news_item&px=Xorg-IGLX-Potential-Bye-Bye
- https://wiki.archlinux.org/index.php/VirtualGL
- https://cdn.rawgit.com/VirtualGL/virtualgl/2.5/doc/index.html
- https://en.wikipedia.org/wiki/Direct_Rendering_Infrastructure
- https://unix.stackexchange.com/a/1441

CHAPTER

NINE

DEPLOYING AMS IN THE CLOUD USING AWS PARALLELCLUSTER

9.1 Introduction

AMS can run on most modern Linux platforms, which includes cloud servers. In this guide we will demonstrate how to set up a basic SLURM cluster on AWS, install AMS on it, and configure SLURM queues in the AMS GUI.

Several cloud providers have supported tools for setting up a SLURM cluster in the cloud (https://slurm.schedmd.com/elastic_computing.html). This provides a useful mechanism for running batch workloads in a cloud environment, without need for much additional setup. It will allow you to use the powerful scheduling capabilities of SLURM in a setting where compute nodes are ephemeral and dynamically scaled up and down with demand to save on costs. AMSJobs, and PLAMS have integrated support for submitting jobs to a SLURM queue.

If you are interested in setting up an HPC cluster in the cloud, you could consider options from major cloud providers such as

- AWS ParallelCluster (https://aws.amazon.com/hpc/parallelcluster/)
- Azure CycleCloud (https://docs.microsoft.com/en-us/azure/cyclecloud/?view=cyclecloud-8/)
- Using SLURM for Google Cloud (https://cloud.google.com/blog/products/gcp/easy-hpc-clusters-on-gcp-with-slurm/)
- HPC Cluster by Oracle (https://cloudmarketplace.oracle.com/marketplace/en_US/listing/67628143)

In this guide we demonstrate how to use AMS on a Linux SLURM cluster on Amazon EC2, configured using AWS ParallelCluster. AWS ParallelCluster uses a relatively simple configuration file to set up a complicated CloudFormation stack (https://aws.amazon.com/cloudformation/) capable of providing an automatically scaling HPC cluster, fully hosted on AWS EC2. The basic steps and some recommendations will be covered by this guide, but for advanced configurations we recommend checking the AWS documentation.

This guide is divided into the following steps.

1. Installing and configuring the cluster
    - We will cover some of the basic steps, and some configuration options that we recommend when running AMS.
2. Installing and setting up AMS on the cluster
    - Where to install AMS, and how to set up the environment so that it runs correctly.
3. Configuring your local installation of AMS to connect to the cluster
    - Setting up remote queues the correct way to use the SLURM cluster

The last part of the guide also will use an installation of AMS on your local machine, but it is not a prerequisite for using AMS on ParallelCluster.

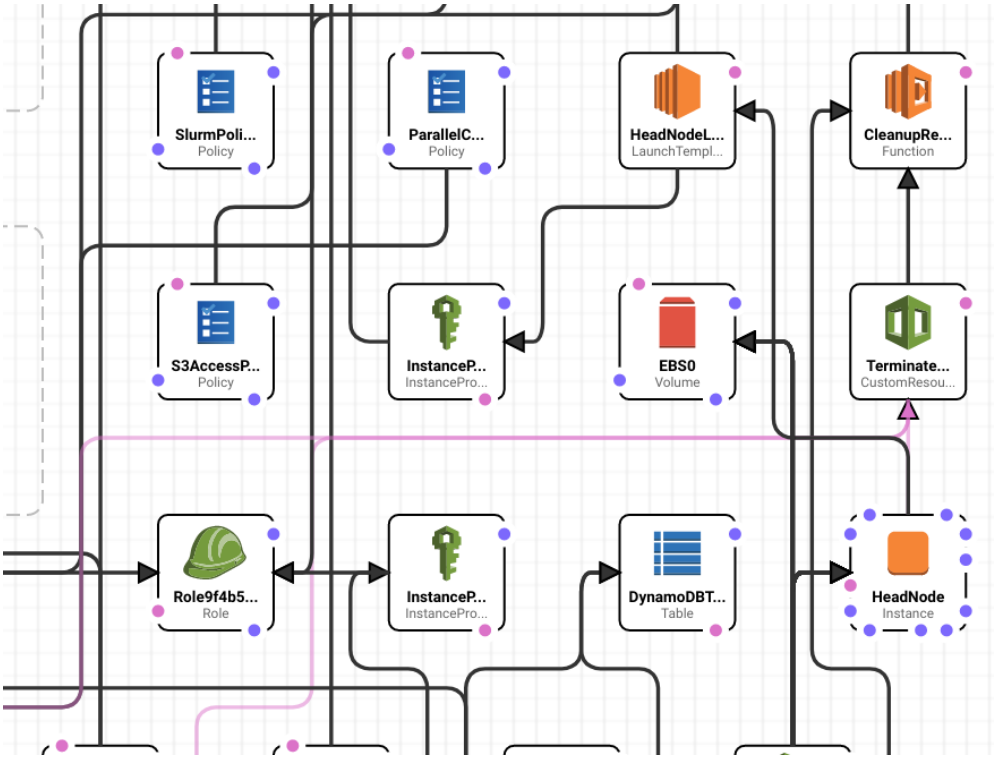


Fig. 9.1: AWS ParallelCluster sets up a CloudFormation stack to deploy all the resources you need to run your cluster on AWS automatically, and easily.

9.2 Prerequisites

The following guide presumes that you have:

- An AWS account, and the AWS command line tools (https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html) installed
- Familiarity with command line use, python, shell scripting, and SLURM.
- Installed python 3.9 or greater, and a recent version of Node.js (no javascript knowledge required)
- Some basic experience with AWS services such as EC2, S3, and IAM, and knowledge about good security practices
- An SSH key pair for EC2 (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html)
- A cloud license for the version of AMS you wish to install. For questions regarding licenses, please email license@scm.com.

During this guide we will use several AWS services. It is important that your AWS account has permission to use these services. Make sure you have the right IAM permissions (https://docs.aws.amazon.com/parallelcluster/latest/ug/iam-roles-in-parallelcluster-v3.html) in order to successfully deploy the cluster.

9.3 Setting up your cluster with AWS ParallelCluster

Note: For the latest instructions and options for ParallelCluster, please check the AWS documentation (https://docs.aws.amazon.com/parallelcluster/latest/ug/what-is-aws-parallelcluster.html)

- If you have never set up a cluster on AWS parallelcluster before, this section will guide you through the process and will provide good starting point to run AMS.
- If you already have a working SLURM cluster running on AWS, you can skim it for recommendations and continue to *Installing and configuring AMS* (page 63).

9.3.1 Installing ParallelCluster

AWS ParallelCluster is a command line tool that helps you deploy a dynamically scaling compute cluster on AWS. It provides a fairly simple way to run batch workloads in the cloud, also allowing software such as AMS to employ existing SLURM features.

1. Install the command-line tool using the instructions provided by AWS (https://docs.aws.amazon.com/parallelcluster/latest/ug/install-v3-parallelcluster.html).

This guide is written for AWS ParallelCluster version 3. Note that you will require a recent version of python 3 (3.9 or greater), and a recent version of Node JS. These are dependencies of ParallelCluster. You will not need to write python or JS code as part of the setup. We highly recommend that you install ParallelCluster using python's virtualenv (https://docs.aws.amazon.com/parallelcluster/latest/ug/install-v3-virtual-environment.html). Keeping it inside a separate virtual environment will prevent other pip installations from breaking the tool, to make sure you can always manage any clusters that you may have deployed.

2. After installing ParallelCluster, make sure you can run it successfully with the following command:

```
(apc-ve) user@mymachine:~/aws$ pcluster version
{
  "version": "3.13.2"
}
```

After installing, we can configure a basic cluster.

9.3.2 Configuring ParallelCluster

Before you configure, make sure you already created an EC2 SSH key pair (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html), as you will be asked to provide it. For this tutorial you will need an `ed25519` format SSH key pair. This is a security requirement for using the most recent Ubuntu images.

The configure command will run interactively, asking you to provide the basic information necessary to create your cluster. For more information, see the instructions provided by AWS (https://docs.aws.amazon.com/parallelcluster/latest/ug/install-v3-configuring.html). It is recommended to use this

to create a template to work with, as it will automatically help you set up your cluster's network, and select some other basic options. Here, we will tell you which recommended options to pick for the purpose of this guide.

1. Run the following configure command.
using the `--config` flag, you can select a different name for the configuration file if preferred.

```
(apc-ve) user@mymachine:~/aws$ pcluster configure --config cluster-config.yaml

INFO: Configuration file cluster-config.yaml will be written.
Press CTRL-C to interrupt the procedure.
```

2. Select the region you want to host the cluster in. This may depend on your own location, or the location of resources you want to connect to the cluster.

```
Allowed values for AWS Region ID:
1. ap-northeast-1
2. ap-northeast-2
3. ap-south-1
4. ap-southeast-1
5. ap-southeast-2
6. ca-central-1
7. eu-central-1
8. eu-north-1
9. eu-west-1
10. eu-west-2
11. eu-west-3
12. sa-east-1
13. us-east-1
14. us-east-2
15. us-west-1
16. us-west-2
AWS Region ID [eu-central-1]:
```

3. Select the key name you want to use for connecting to the cluster.

```
Allowed values for EC2 Key Pair Name:
1. mykey
2. otherkey
EC2 Key Pair Name [mykey]: 1
```

Make sure you have this key available on the machine you will use to connect to the cluster, and that it is an ed25519 format key. Next, you will need to select a scheduler.

4. Please select **SLURM** as the scheduler.

```
Allowed values for Scheduler:
1. slurm
2. awsbatch
Scheduler [slurm]: 1
```

5. Please select **ubuntu2404** as the operating system.

```
Allowed values for Operating System:
1. alinux2
2. alinux2023
3. ubuntu2004
4. ubuntu2204
5. ubuntu2404
6. rhel8
7. rocky8
8. rhel9
9. rocky9
Operating System [alinux2]: 5
```

We choose the Ubuntu 24.04 Operating System here as an example.

Next, we need to select a head node instance type. Even though you should not be running compute jobs on the head node, you will still need a decent amount of processing power to run the SLURM cluster and manage the network. You should also make sure to select an instance type that has a decent network bandwidth, since most network connections are routed through the head node.

6. Select a head node type. For this guide we are using a `c5.2xlarge`.

```
Head node instance type [t2.micro]: c5.2xlarge
```

In the next step, a compute queue is defined. This queue contains a set of compute resources (EC2 instance types), and some configuration options for each resource.

7. Set up 1 queue. Call it `queue1`
For this guide we will set up one queue at this step.

```
Number of queues [1]: 1
Name of queue 1 [queue1]: queue1
```

8. For this queue, please choose **1 compute resource**, and use **c5n.xlarge** as instance type. Leave the maximum at 10.

```
Number of compute resources for queue1 [1]:
Compute instance type for compute resource 1 in queue1 [t2.micro]: c5n.xlarge
Maximum instance count [10]:
```

The number of compute resources refers to the *number of different instance types*. As an example, we are using only `c5n.xlarge` instances, with a Maximum count of 10. These are Intel based, and have 2 cores with 8 GB of memory. When choosing instance types please note that the number of cores typically is only half the number of vCPUs. For optimal use of resources disabling simultaneous multithreading (known as hyper-threading on Intel platforms) is recommended, so that only one thread runs per core.

Next up is the network configurations.

9. Select a VPC, or automate creation of a new one.
If you already have a Virtual Private Cloud (VPC), you can use the one that already exists.
If you are unsure which to use or don't have a VPC, the configure tool can generate a VPC for you.

```
Automate VPC creation? (y/n) [n]:
Allowed values for VPC ID:
  #  id            name      number_of_subnets
---  ------------  ------  -------------------
  1  vpc-3213910c  vpc1                      5
VPC ID [vpc-3213910c]: 1
```

For more information about VPCs on AWS, please read the AWS documentation (https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html) on the subject. Next up, you will need to define subnets both for the head node and the compute nodes. Automating this is easy, select an availability zone and configuration.

10. For this guide **Automate subnet creation**.
For simplicity and to save costs, we use **2. Head node and compute fleet in the same public subnet** .

For production you may prefer a private subnet. Note that using a private subnet will create a NAT gateway (https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html), which could incur some extra costs.

```
Automate Subnet creation? (y/n) [y]:
Allowed values for Availability Zone:
1. eu-central-1a
2. eu-central-1b
3. eu-central-1c
Availability Zone [eu-central-1a]: 1
Allowed values for Network Configuration:
1. Head node in a public subnet and compute fleet in a private subnet
2. Head node and compute fleet in the same public subnet
Network Configuration [Head node in a public subnet and compute fleet in a private␣
↪subnet]: 2
```

That should be the last step in the configuration. Next, a CloudFormation stack will be created for your network configuration. Note that this is different from the stack that will be used to deploy the cluster. That stack won't be created until later.

11. Wait until the stack generation is finished.

```
Creating CloudFormation stack...
Do not leave the terminal until the process has finished.
Stack Name: parallelclusternetworking-pubpriv-20210923123456 (id:␣
↪arn:aws:cloudformation:eu-central-1:112233445577:stack/parallelclusternetworking-
↪pubpriv-20210923123456/21615d90-2059-11ec-b5c3-02cac9daa27a)
Status: parallelclusternetworking-pubpriv-20210923123456 - CREATE_COMPLETE
The stack has been created.
Configuration file written to cluster-config.yaml
You can edit your configuration file or simply run 'pcluster create-cluster --cluster-
↪configuration cluster-config.yaml --cluster-name cluster-name --region eu-central-1
↪' to create your cluster.
```

At this point, you have a basic configuration file from which you could launch a cluster. However, it will need some editing to add some options that will make it easier to use AMS effectively.

To continue configuring, **open the configuration file in a text editor**. In this example we called the file `cluster-config.yaml`. Your basic cluster file will look something like this:

```
Region: eu-central-1
Image:
  Os: ubuntu2404
HeadNode:
  InstanceType: c5.2xlarge
  Networking:
    SubnetId: subnet-0bd780a5354103
  Ssh:
    KeyName: mykey
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - Name: c5nxlarge
      InstanceType: c5n.xlarge
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-0bd780a5354103
```

In the next few subsections, we will provide recommendations for a lot of the individual options. These will allow you to use AMS more effectively on your cluster. If something is optional, it will be indicated as such.

9.3.3 Defining queues

The configuration tool will have guided you through the definition of at least one compute queue for use. Inside the configuration file you will find the `Scheduling` section, and within the `SlurmQueues` section.

Under the `ComputeResources` section, you can define different types of compute nodes. Each compute resource will require an instance type and a name. You can also setup a minimal and maximal number of instances.

One setting you should add to your compute resource is the `DisableSimultaneousMultithreading: true`, which will disable hyper-threading on your compute nodes.

Please Add **DisableSimultaneousMultithreading: true** key to your compute resource.

Here is an example of a queue setup with one on-demand queue, and hyper-threading disabled on its compute resource. You can add as many as 10 queues, and 5 compute resources per queue.

```
SlurmQueues:
- Name: queue1
  CapacityType: ONDEMAND
  ComputeResources:
  - Name: c5nxlarge
    InstanceType: c5n.xlarge
    DisableSimultaneousMultithreading: true
  Networking:
    SubnetIds:
    - subnet-0bd780a5354103
```

Tip: Internode communication speed is crucial for running MPI jobs across multiple devices. If you wish to run multinode jobs effectively you should look into using the Elastic Fabric Adapter (EFA) (https://docs.aws.amazon.com/parallelcluster/latest/ug/Scheduling-v3.html#yaml-Scheduling-SlurmQueues-ComputeResources-Efa) option in your configuration. This will limit your choice of instance types, but allow for much faster multinode jobs.

In this guide, we will not use Elastic Fabric Adapter (EFA). Therefore, the configuration presented here is mostly useful for single-node jobs.

9.3.4 Adding a shared directory for software installations and jobs

In order to provide AMS (and potentially other software) on your cluster, you need a shared filesystem that will be present on both head and compute nodes. For this you use the `SharedStorage` option, which is present on the top level of the configuration file. ParallelCluster provides several options, such as Elastic Block Store (EBS) (https://aws.amazon.com/ebs/), Elastic File System (EFS) (https://aws.amazon.com/efs/), and FSx for Lustre (https://aws.amazon.com/fsx/lustre/).

To keep things simple, here we define a single 64Gb shared EBS filesystem that uses gp2 SSD storage for installing software and storing job data.

Please add the following **SharedStorage** section to your configuration file.

```
SharedStorage:
- Name: ebs
  MountDir: /shared-ebs
  StorageType: Ebs
  EbsSettings:
    VolumeType: gp2
    Size: 64
    Encrypted: false
    DeletionPolicy: Delete
```

In production you will probably have other needs, but for this guide this will suffice. You may find that for your workflows it could be useful to have a high-performance EBS scratch directory, or a Lustre filesystem. It may also be useful to store job results on a separate, cheaper filesystem, to make sure enough space is available. In the cloud, access to filesystems could be limited by the network performance, and that in a typical case this network access is routed through the head node. Therefore if you experience IO slowdowns, it is also important to check the network load on the head node, and make sure you have selected an instance type that has high available bandwidth.

Warning: **Newly defined storage systems** in the cluster configuration will be **deleted** when the cluster is deleted, unless you change the default deletion policy. If you delete a cluster, be sure not to lose any important data! This does not apply to pre-existent storage systems that are mounted.

9.3.5 Optional: Security groups

Your cluster should only be accessible by those that have the private key that you provided. By default, ParallelCluster will configure a security group that allows access from all IPv4 addresses (0.0.0.0/0). For increased security, you may also like to restrict access to known IP-addresses. In the `Ssh` section, you will already find the `KeyName` that points to the ssh key you provided. You can add the additional `AllowedIps` key to this section, and specify ip ranges using CIDR notation. Below here is an example of how to restrict a specific ip address, or a range of addresses.

```
HeadNode:
  ....
  Ssh:
    KeyName: mykey
    AllowedIps:
     - 12.34.123.0/24 # all office machine
     - 12.34.124.123/32 # specific machine
```

9.3.6 Optional: GUI access with DCV

AWS ParallelCluster can install a DCV server (https://www.amazondcv.com/) on your head node, in order to provide a graphical environment. In the already defined `HeadNode` section,add a new `Dcv` section, and a key `Enabled: True` (see also the example below). You can also set the `AllowedIps` using CIDR notation to restrict access to specified IP ranges.

```
HeadNode:
  ....
  Dcv:
    Enabled: true
    AllowedIps: 12.34.123.234/32
```

Note that to use the GUI for AMS you will also need to install GUI dependencies. On a recent Ubuntu system, you can use apt to install packages. The default packages on the AWS parallelcluster image for Ubuntu 24.04 should be sufficient to run the AMS GUI.

You may also need to run the GUI in software openGL mode, by setting `SCM_OPENGL_SOFTWARE=1` in your environment before starting up GUI programs.

9.3.7 Creating the cluster

This concludes the basic configuration of the cluster. At this point you should have finished creating your cluster configuration as a yaml file. Make sure you have taken a little time to read the best practices (https://docs.aws.amazon.com/parallelcluster/latest/ug/best-practices-v3.html) as well.

Next, the cluster can be started. In order to start the cluster, you need to use `pcluster create-cluster`. This will deploy a CloudFormation stack that contains all your cluster components. You need to provide a cluster name, and the path to your configuration file. pcluster will validate your configuration file first, and if valid it will start your cluster.

Start the cluster using the **create-cluster** subcommand

```
pcluster create-cluster --cluster-name ams-cluster --cluster-configuration cluster-
↪config.yaml
```

If you run into any errors with the configuration file, look if any manually added keys are in the right section of the file and that you used correct YAML syntax. A full specification of the configuration format is available online (https://docs.aws.amazon.com/parallelcluster/latest/ug/cluster-configuration-file-v3.html).

The cluster will be created through a CloudFormation stack which has the same name as the cluster. This will take a while. You can follow progress through the CloudFormation console (https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-console.html). If any errors occur, you can see them there. Common errors may include a lack of permissions to create resources. You can also follow progress on the command-line with the `pcluster describe-cluster` command.

Follow progress using the following command

```
pcluster describe-cluster --region eu-central-1 --cluster-name ams-cluster
```

If your cluster fails to create due to permission errors, please talk to your AWS account administrator about IAM permissions. After it is done, it will list `"clusterStatus": "CREATE_COMPLETE"`.

9.3.8 Connecting to the head node

If your cluster has reached `CREATE_COMPLETE` status, you should now be able to ssh to your cluster. An easy way to do this is using

```
pcluster ssh --region eu-central-1 --cluster-name ams-cluster -i ~/.ssh/mykey.pem
```

Alternatively, you can look up the head node in the EC2 dashboard (https://console.aws.amazon.com/ec2/v2/home).

If you opted to install a DCV server, you can also connect with NiceDCV. An easy, one line way to connect is

```
pcluster dcv-connect --region eu-central-1 --cluster-name ams-cluster --key-path ~/.
↪ssh/mykey.pem
```

Next you can install AMS on the cluster.

9.3.9 Deleting the cluster

If in the future you wish to shutdown the cluster, you can use the `pcluster delete-cluster` command.

```
pcluster delete-cluster --region eu-central-1 --cluster-name ams-cluster
```

Keep in mind that storage systems containing your data may be deleted if they were newly defined through the cluster configuration file. If you wish to change this behavior, you can check the information about shared storage in the AWS documentation (https://docs.aws.amazon.com/parallelcluster/latest/ug/SharedStorage-v3.html).

9.4 Installing and configuring AMS

You can follow the basic instructions in the installation guide, while taking a few caveats in mind. We will summarize the main steps here for convenience, and help you correctly use AMS with IntelMPI and SLURM.

Since you need access to AMS on the compute nodes, it is important to install it on a filesystem that is shared between the compute and head nodes. In the cluster configuration we defined the `shared-ebs` EBS filesystem. Please go into that directory, and download and extract AMS there.

1. Download and extract the AMS tarball.

```
cd /shared-ebs
URL=https://downloads.scm.com/Downloads/download2025/bin/ams2025.104.pc64_linux.
↪intelmpi.bin.tgz
CREDS=u12345:myscmpassword
curl -u "$CREDS" -sL "$URL" | tar xz
```

If you have configured a different storage location, extract the tarball in a folder there. This command downloads and extract the binaries in a single step by using curl and tar. You will have to provide your SCM login details. Depending on which instance types you use, always use an appropriate set of binaries to match your CPU types. Here, we opted to use the AMS binaries that are optimized for Intel CPUs. If you use AMD-Zen based instances (https://aws.amazon.com/ec2/amd/), you will want to download the version of AMS compiled with optimizations for AMD.

You will also have to set up the environment, both on your head and compute node. Inside the AMS installation folder (typically named something like `ams2025.104`) you will find the `Utils` folder. There, you can find a template `amsrc.sh` script, but it will require some editing.

2. Make a copy of the `$AMSHOME/Utils/amsrc.sh` script to somewhere on your shared filesystem, for instance in the top directory of the installation.

To distinguish it from other scripts such as `$AMSHOME/amsbashrc.sh` (which you should not use on AWS), you could call it something like `cloudrc.sh`.

```
cd /shared-ebs/ams2025.104
cp Utils/amsrc.sh ./cloudrc.sh
```

Edit the `AMSHOME` variable to point to the top folder containing your installation on the shared filesystem.

3. `AMSHOME=/shared-ebs/ams2025.104`

If in your configuration you have defined a high performance scratch filesystem, you can set the `SCM_TMPDIR` variable to point to it as well. You may wish to set the `SCM_PYTHONDIR` to somewhere in the shared filesystem. That way if you need to install any additional python packages, you only have to do it once.

Remove the `SCMLICENSE` line, as it is not used in cloud environments. Instead, add a line with your `SCM_CLOUD_CREDS` which will allow the software to locate your license over the internet. Don't forget to export this additional variable. Save the file when you're done.

4. Delete the `SCMLICENSE` line
5. Add a line `export SCM_CLOUD_CREDS=u12345:myscmpassword`
6. Save the file.
7. Make sure you can source the file without error, using `source cloudrc.sh`.

Remember that before running any jobs, you will need to source this file. You can add it to your `prolog` command in AMSJobs remote queue command, which will be covered later on. Next, we need to check if your license works correctly.

8. run the `dirac TARGET`, to check that the cloud license works for your target.

For instance if you want to use ADF, call the `dirac ADF` command. If this returns an error check your credentials carefully. Also check the spelling of the target which is case sensitive. If you don't have a valid license you will see a message such as "Your license does not include module ADF version 2025.104 on this machine". For information and support regarding licenses please contact license@scm.com.

9.4.1 Configuring IntelMPI and SLURM

AMS is capable of running jobs in parallel, both across processor cores and several instances by using IntelMPI. In order to use AMS with both IntelMPI with SLURM, some configuration is needed. Specifically, the `I_MPI_PMI_LIBRARY` environment variable needs to point to a valid path, containing the file `libpmi.so`.

On the AWS parallelcluster, you can find libpmi.so under `/opt/slurm/lib/libpmi.so`.

1. Edit your `cloudrc.sh` script, add a line that says
`export I_MPI_PMI_LIBRARY=/opt/slurm/lib/libpmi.so`

In order to make best use of IntelMPI, you will have to configure AMS to use the version of IntelMPI that AWS parallelcluster preinstalls for you.

First, to use the IntelMPI installation that comes installed on the AWS cluster you will have to load the intel mpi module. This needs to be done before every job.

```
module load intelmpi
```

You will also need to `export SCM_USE_LOCAL_IMPI=1` inside your environment at run time. To make it simpler, you can put both the export, and module load commands inside the **cloudrc.sh** script.

2. add a new line to the `cloudrc.sh` script created earlier:
`module load intelmpi`
3. Add another line saying
`export SCM_USE_LOCAL_IMPI=1`
, to tell AMS to use the Intel MPI version installed on the cluster itself. Save and close the file.

You should now be ready to run AMS jobs in parallel on your cluster.

Your final `cloudrc.sh` may look something like the example below:

```
#!/bin/sh
# This file should be sourced by a bourne shell (sh), bash shell (bash), or z-shell␣
↪(zsh).

# AMSHOME refers to the folder that contains the folders "bin", "atomicdata",
↪"scripting" and a few others
AMSHOME=/shared-ebs/ams2025.104
# SCM_TMPDIR is used for writing temporary data during calculations. For best␣
↪performance, this needs to be fast&local storage (no network mount).
SCM_TMPDIR=/tmp

# SCM_PYTHONDIR should point to the location where you want the SCM python stack to␣
↪set up the virtual environment.
# This is for installing additional python packages from PyPi with "amspython -m pip␣
↪install"
SCM_PYTHONDIR=/shared-ebs/ams2025.104/Python

# SCM_CLOUD_CREDS should contain your license credentials from SCM.
SCM_CLOUD_CREDS=u12345:password

# MPI settings for AWS ParallelCluster
module load intelmpi
export I_MPI_PMI_LIBRARY=/opt/slurm/lib/libpmi.so
export SCM_USE_LOCAL_IMPI=1

# Variables below this line usually do not need to be changed
AMSBIN="$AMSHOME"/bin
AMSRESOURCES="$AMSHOME"/atomicdata

export AMSHOME AMSBIN AMSRESOURCES SCM_TMPDIR SCM_CLOUD_CREDS SCM_PYTHONDIR

# add $AMSBIN to the PATH
PATH="$AMSBIN":"$PATH"
export PATH
```

9.5 Using the queues on your local AMS installation

In this section, it will be explained how to use queues both through your local installation of AMS, and by running on the head node.

To use your cluster from a local installation of AMS, in principle you just need to follow the instructions for AMSJobs.

From AMSJobs select *Queue* >> *New* >> *SLURM*.

- `Queue name`: Give the queue a name which will be used inside AMSjobs to refer to it, for instance *pcluster queue1*.
- `Default options`: leave it at `1`
- `Remote host`: Fill in the public Ipv4 address of your cluster's Head Node. You can find this in the EC2 console. Alternatively, you can use the `pcluster ssh` command to connect, as it will also tell you the hostname when you log in.
- `Remote user`: The username should be `ubuntu` for Ubuntu systems, for other distributions it is `ec2-user`.
- The `Remote job directory` should point to somewhere on a shared filesystem, such as a `shared-ebs/jobs`. The GUI will copy jobs to the head node.

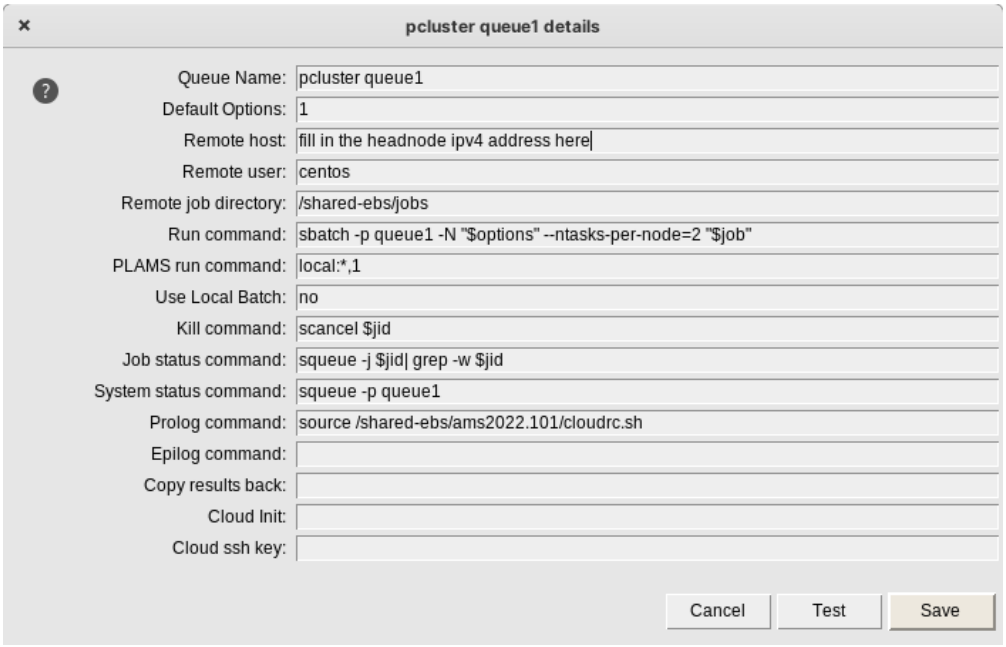


Fig. 9.2: Configuring your queue in AMSjobs requires you to fill in a few options.

- `Run command`: Jobs should be submitted using `sbatch` command. Next, some options are needed
  - `-p queuename`, The name of your queue, such as `queue1`, should be entered here.
  - the `-N` flag should be used for specifying how many nodes.
  - the `--ntasks-per-node=X` flag should be specified to match the number of cores (not vCPUs!) on the instance type.
  - If you want to mix different instance types in one queue, and request a specific instance type, use the `-C flag.` for instance, `-N 3 -C "[c5.xlarge*1&t2.micro*2]"` will give you three instances, one `c5.xlarge` instance, and two `t2.micro` instances
- `Use local batch` no.
- `Kill command`: Use the default setting (`scancel $jid`)
- `Job status command`: Use the default setting (`squeue -j $jid| grep -w $jid`)
- `System status command`: Add the name of your queue. For example `squeue -p queue1`.
- `Prolog command`: Source your cloudrc.sh script here:
  - `source /shared-ebs/ams2025.104/cloudrc.sh`
- `Cloud ssh key`: Leave this empty. You should use an SSH agent to provide your SSH key to AMSjobs. The "path to private key" field is not supported for SLURM configurations. On Linux/MacOS you can use ssh-add, or create a configuration in your `$HOME/.ssh/config` file that uses your AWS key file. On Windows you can use pageant. Please see the detailed SSH instructions in the installation manual for more information.

Other fields you can also leave empty.

Save the queue. After that, you can use the queue to submit jobs.

9.6 Considerations

The configuration options provided in this guide are a simple and easy way to get started. We provided a configuration that is mostly useful for those running single-node jobs, and only mount a small filesystem. In production you will find that you need different options for filesystems, compute instance types and network configuration.

If you want to run multi-node jobs, consider using Elastic Fabric Adapter (https://docs.aws.amazon.com/parallelcluster/latest/ug/Scheduling-v3.html#yaml-Scheduling-SlurmQueues-ComputeResources-Efa).

If your cluster will be active for a long time, and you wish to write many jobs, consider using a proper file system like an elastic file system (EFS) (https://aws.amazon.com/efs/) , FsX for Lustre filesystem (https://aws.amazon.com/fsx/lustre/).

There are many more options available. Please see the official AWS documentation (https://docs.aws.amazon.com/parallelcluster/latest/ug/cluster-configuration-file-v3.html) for an up to date overview.

9.7 Appendix

9.7.1 Monitoring the cluster

AWS ParallelCluster will create a CloudWatch dashboard (https://aws.amazon.com/cloudwatch/) to monitor your cluster. There you can see information such as CPU, disk, and network usage.

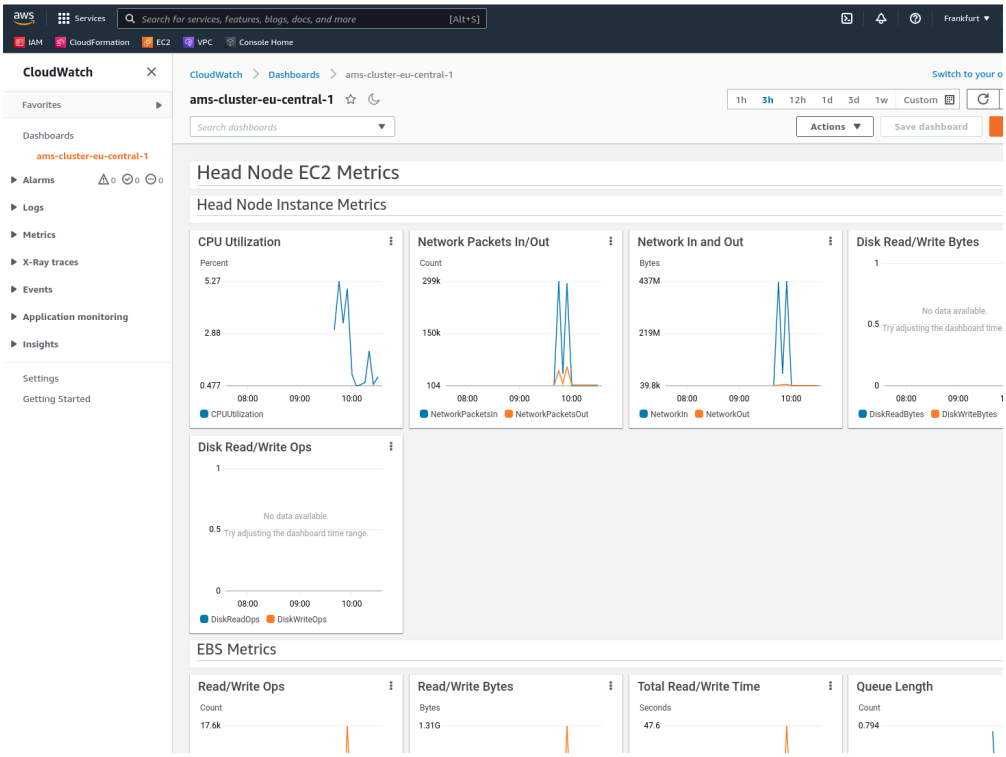


Fig. 9.3: The cloudwatch dashboard visualizes useful system information about your cluster.

9.7.2 Debugging issues

ParallelCluster uses CloudFormation to deploy all the different AWS resources that are needed. If something goes wrong during some stage of the deployment, it will automatically start rolling back changes. This can make it difficult to debug sometimes. To aid in debugging, provide the `--rollback-on-failure` to the `pcluster create-cluster` command. This will prevent the rollback procedure. You can then use the Cloudformation console (https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-console.html) to inspect your stack, and see what went wrong.

9.7.3 An example cluster

Here is an example of a fully configured cluster. It defines both an on-demand, and a spot queue. The head node has DCV installed, and SSH and remote GUI access are restricted to pre-configured IP-addresses. The head node is configured with a custom bootstrap script present in an S3 bucket, and the cluster has read-only access to said bucket. The bootstrap script installs AMS, and also installs GUI dependencies on the head node.

```
Region: eu-central-1
Image:
Os: ubuntu2404
HeadNode:
  InstanceType: c5n.xlarge
  Networking:
    SubnetId: subnet-77daea63d414c9c4
  Ssh:
    KeyName: mykey
    AllowedIps: 12.34.123.234/32
  Dcv:
    Enabled: true
    AllowedIps: 12.34.123.234/32
  CustomActions:
    OnNodeConfigured:
      Script: s3://cluster-utils/post-install.sh
      Args:
        - INSTALL_AMS=YES
        - INSTALL_GUI_DEPS=YES
Iam:
  S3Access:
    - BucketName: cluster-utils
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    CapacityType: ONDEMAND
    ComputeResources:
    - Name: c5nxlarge
      InstanceType: c5n.xlarge
      DisableSimultaneousMultithreading: true
      MinCount: 0
      MaxCount: 10
    - Name: c4xlarge
      InstanceType: c4.xlarge
      DisableSimultaneousMultithreading: true
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
```

(continues on next page)

(continued from previous page)

```
      - subnet-5208860694ca8109
SharedStorage:
- Name: ebs
  MountDir: /shared-ebs
  StorageType: Ebs
  EbsSettings:
    VolumeType: gp2
    Size: 64
    Encrypted: false
    DeletionPolicy: Delete
```

Note that you should of course use your own, preconfigured subnets and information such as ip addresses and S3 buckets.

9.7.4 Recommended reading

- The AMS installation guide covers all the basics about installation.
- The AMSjobs user manual describes in detail how to set up a queue.

CHAPTER

TEN

APPENDIX A. ENVIRONMENT VARIABLES

If you start the MacOSX AMS-GUI application the environment defined by your shell startup scripts is ignored. Instead the bundled AMS is used, and environment variables may be defined in a file $HOME/.scmenv .

The following environment variables must always be set for all AMS versions.

AMSHOME: full path of the AMS installation directory, for example, $HOME/ams2021.101 .

AMSBIN: full path AMS directory with binaries, typically set to $AMSHOME/bin .

AMSRESOURCES: full path of the directory with the AMS database (basis sets and so on), typically set to $AMSHOME/atomicdata

SCMLICENSE: full path-name of the license file, for example $AMSHOME/license.txt.

SCM_DOMAINCHECK: set to yes if you have a license based on your domain info (DNS). If it is defined but no proper DNS sever is available, delays will often occur.

SCM_TMPDIR: full path of the directory where all processes will create their temporary files. See also the Installation manual and the special section on SCM_TMPDIR below.

The following environment variable may be required at run-time by a parallel version.

NSCM: The number of processes to be used in a particular calculation. This variable should only be set per job. Do **not** add any NSCM definitions to your shell resource files.

NSCM_AMSEXTERNAL: Number of processes used by external engines called by the AMS driver. Used by engines like MLPotential or Quantum ESPRESSO to configure their parallelization separately from the AMS driver. When SCM_DISABLE_MPI=1 is set, this variable by default inherits the current value of NSCM, so that the AMS driver runs in serial and the engine runs with NSCM processes.

SCM_DISABLE_MPI: If defined, sets the number of processes to be used (NSCM) to 1 and forces the direct execution of the program, i.e. not going through mpirun or the corresponding launcher of the batch system (e.g. srun for Slurm). This may be required when the AMS driver is used with an external or or pipe based engine (e.g. MLPotential or Quantum ESPRESSO) under a batch system to avoid recursive mpirun calls.

SCM_MACHINEFILE full path-name of the file containing a list nodes to use for a job; **Important:** this variable should **only** be set if multiple computers are used without any batch system and then it should be set on the per-job basis. The file pointed to by the variable must already exist and it must contain a list of nodes on which a particular job is about to be executed, possibly with their processor count.

SCM_USE_LOCAL_IMPI this applies only to IntelMPI distributions. Setting this environment variable to not be empty will disable the IntelMPI runtime environment shipped with the AMS distribution, allowing the use of a local IntelMPI installation, or any other ABI-compatible local MPI installation (MPICH v3.1 or newer for example). The environment must be properly set up on the machine, meaning I_MPI_ROOT must be set, and mpirun should be in the PATH, and the libraries must be in LD_LIBRARY_PATH.

SCM_USE_IMPI_2021 this applies only to IntelMPI distributions of AMS2021 and newer. Setting this environment variable to not be empty will load the IntelMPI 2021 runtime shipped with the AMS distribution instead of the 2018 version. This can solve problems on certain linux distributions, cluster environments and/or AMD Zen processors.

SCM_USE_LOCAL_OMPI this applies only to OpenMPI distributions. Setting this environment variable to not be empty will disable the OpenMPI runtime environment shipped with the AMS distribution, allowing the use of a local OpenMPI 4.1.1 installation. The environment must be properly set up on the machine, meaning OPAL_PREFIX must be set, and mpirun should be in the PATH, and the libraries must be in LD_LIBRARY_PATH.

SCM_MPIRUN_EXE if this environment variable is set, AMS will use it as the executable to launch the MPI job with. Make sure to also set SCM_MPIRUN_OPTIONS when using this! By default this variable gets filled in automatically in the $AMSBIN/start script, based on the detected environment.

SCM_MPIRUN_OPTIONS if this environment variable is set, AMS will use it as arguments to the MPI launch command. If you only want to add arguments to the MPI execution instead of overruling the default arguments set by AMS, use SCM_MPIOPTIONS instead!

SCM_SRUN_OPTIONS gets added to the launch command if AMS is running under Slurm and SCM_MPIRUN_EXE is set (or automatically determined) to be the srun. You can use this environment variable to add Slurm specific flags to the launch command, while making sure these options are only used if the job is actually running under Slurm.

SCM_MPIOPTIONS anything added to this environment variable gets added to the MPI launch arguments. This variable does not influence the autodetect scheme for finding the right mpirun commands.

SCM_PYTHONDIR: This environment variable specifies where the AMS python distribution will search for a python virtual environment to use. If no suitable virtual environment exists in this directory, it will be created. To not use a virtual environment, set SCM_PYTHONDIR to an empty string.

SCM_PYTHON_VENV_NO_BRANCH: Set this environment variable to a non-empty value to force `amspython` to name the virtual environment without the branch suffix. It also makes AMSpackages generate a branch-independent default package directory name when AMSpackages is configured. This is mainly useful for AMS developers that want to have separate virtual environments per checkout or git worktree, instead of per branch.

SCM_PYTHONPATH: The contents of this environment variable gets prepended to the PYTHONPATH variable when using the AMS python distribution. You can add the paths to other python modules that you wish to use in combination with the shipped python distribution to this variable to make them available.

The following environment variables are relevant for the GUI modules. For a full list, see the GUI Environment Variables page.

SCM_GUIRC: location of the preferences file, by default $HOME/.scm_guirc

SCM_GUIPREFSDIR: location of the preferences folder, by default $HOME/.scm_gui/ (available since ADF2013.01b)

SCM_TPLDIR: location of the templates directory, by default no extra templates are loaded

SCM_STRUCTURES: location of the structures directory, by default no extra structures are loaded

SCM_RESULTDIR: location of the results directory, by default the current directory used

DISPLAY: specifies the X-window display to use and is required for all X11 programs on Linux/Unix and Mac OS X. On MaC OS X you should typically not set it as it will be set automatically. Setting it will break this.

SCM_QUEUES: path to the dynamic queues directory, by default AMSjobs will search the remote $HOME/.scm_gui file

SCM_OPENGL_SOFTWARE: Linux and Windows (available since AMS2020). If set to be non-empty, the GUI will use a software OpenGL implementation. See Using the GUI on a remote machine for more information.

SCM_OPENGL1_FALLBACK: No longer used since AMS2019.305 (applies from ADF2017 until AMS2019.304), see SCM_OPENGL_SOFTWARE and Using the GUI on a remote machine for more information.

The AMS driver environment variables can be found in the AMS Documentation

The following environment variables are relevant for source distributions only, and only at the configure time.

MPIDIR and MATHDIR: see Compiling AMS from Sources

The following environment variables may be set to modify other aspects of AMS execution. All of them are optional and some are used for debugging only.

SCM_GPUENABLED: Environment flag to turn GPU acceleration on or off. Only works for the CUDA-enabled binaries. Setting this variable to TRUE turns GPU acceleration on, setting it to FALSE turns it off. If the input contains a GPU%Enabled input key then this environment variable will be ignored.

SCM_MAXCOMMLENGTH: when performing collective MPI communications AMS breaks them into pieces of at most SCM_MAXCOMMLENGTH elements, each element being 4 or 8 bytes long. So in practice the largest collective MPI operation should not exceed SCM_MAXCOMMLENGTH*8 bytes in size. By default, SCM_MAXCOMMLENGTH is set to 268435455 on Linux and 4194304 on Windows and macOS. When the SCM_MAXCOMMLENGTH environment variable is set its value is used instead.

SCM_VECTORLENGTH: all DFT programs within the AMS package use numerical integration, and this is normally the most time-consuming part of the code. Numerical integration involves summing together results for each 'integration point'. The best performance is achieved when handling a number of points at the same time. The number of integration points handled together is called the block length or the vector length. If the block length is too small, you will have a significant overhead and the programs may become very slow. If the block length is too large, lots of data will not fit in the processor cache and again the program will not run at the maximum speed. The optimal block length is somewhere in between, ranging from 32 to 4096 depending on your hardware. Sometimes it pays off to set the block length explicitly NOT to a power of 2 to avoid memory bank conflicts. Again, try it yourself with your typical calculation on your production machine to find out the optimal value for your situation. On most machines, the default 128 is a good value.

SCM_SHAR_NCORES: setting this variable to a number forces AMS to split each physical node into sub-nodes when allocating shared memory. Each sub-node will contain up to the specified number of processes. This will work only if ranks are assigned to physical and NUMA nodes sequentially. The result is unpredictable if a round-robin rank distribution is used.

SCM_SHAR_PER_SOCKET: setting this variable to any non-empty string forces AMS for Linux to ignore NUMA nodes and use one node per CPU socket instead. This may be helpful when the NUMA granularity is too fine, for example on AMD Zen and other architectures using chiplet design. This variable has no effect when no CPU affinity (or CPU binding) is used.

SCM_SHAR_NONUMA: setting this variable to any non-empty string forces AMS for Linux to ignore NUMA nodes and use the whole shared memory machine as one node. This may be helpful when the job uses too much shared memory otherwise.

SCM_SHAR_EXCEPTIONS: setting this variable to "*" disables the use of shared arrays.

SCM_SHAR_LIMIT: sets the limit on the total size of shared arrays in megabytes. The default is a bit less than half of the node's total RAM. If a new shared array would cause the total amount of shared memory to go over the limit, then instead of placing the array into shared memory it is created as a shared file in the scratch directory of the node-master. The file is then memory-mapped into the address space of all processes on the node. The effect will be the same as when the array is placed into shared memory except that there may be a delay due to disk I/O when the array is destroyed (because on some systems it may have to be written to the disk first).

SCM_KF_PAGESIZE: size increment for KF files in kB (64 by default). Using a large file may increase performance for some distributed file systems that have a large performance penalty on the ftruncate64() system calls. It should be a multiple of 64.

SCM_DEBUG: setting this to a non-empty string will cause each MPI rank to print values of relevant environment variables and some messages about copying files to/from SCM_TMPDIR.

SCM_NOMEMCHECK: setting this to a non-empty string disables checks on memory allocation failures. The usefulness of this variable is questionable.

SCM_TRACETIMER: setting this to a non-empty string will produce additional output on entry/exit to/from internal timers.

SCM_DEBUG_ALL: setting this to yes is equivalent to specifying DEBUG $ALL in the input

10.1 More on the SCM_TMPDIR variable

Below we will explain in more detail how does the SCM_TMPDIR environment work. Every parallel job consists of one master and one or more slave tasks. Master and slaves behave a bit differently with respect to their scratch directories.

Slave processes

Slave processes will always create a directory for their scratch files in $SCM_TMPDIR and *chdir* to it to avoid any risk that shared files are updated by more that one process at the same time. For efficiency reasons, that directory should reside on a local disk unless you are using very, very fast shared file system for large files. You need write access to that directory, and the file system should have enough free space. Please note that the SCM_TMPDIR environment variable will be passed from the master to slaves. After the job is finished, slave processes will delete their scratch directories. This can disabled by setting the SCM_DEBUG environment variable to any text, for example, to "yes". In this case the scratch directory and all its contents will be left intact. This directory will also be left behind when a job has crashed or has been killed. Each slave writes its text output to a file called KidOutput located in its scratch directory. In case of an error this file will likely contain some sensible error message. If an error occurs and a slave process exits in a controllable way then in order to avoid losing the file AMS will copy the file to the directory, from which the job was started, as KidOutput__#, where # is the process' rank.

Master process or serial runs

The master process (which is the only process in a serial run) will also create its temporary files in its own sub-directory of $SCM_TMPDIR. There are some exceptions. Some files, such as logfile and TAPE13, will be created in the directory where AMS was started because they are not performance-critical but are convenient to have in the current directory for different reasons. For example, logfile is nice to have in the current directory in order to follow the calculation progress and the TAPE13 is an emergency restart file that can be used if AMS crashes or is killed. At the end of a calculation, the master will copy all result files from its scratch directory to the directory where it was started.

Using multiple scratch disks

It is possible to use multiple physical scratch disks in a parallel calculation. See the Installation manual for more information about this.

CHAPTER

ELEVEN

APPENDIX B. DIRECTORY STRUCTURE OF THE AMS PACKAGE

Below is the list of major parts of the AMS package.

11.1 The bin directory

When the package is installed, the executable files are placed in $AMSHOME/bin. The binary executable file is usually called 'ams.exe', 'reaxff.exe', and so on. On Windows it is ams.3.exe, reaxff.3.exe, etc. There will also be files called just 'ams' or 'reaxff'. The latter are shell scripts that execute the corresponding binaries.

You should use the script files, rather than the executables directly. The script files provide a correct environment, and if needed prepare for running in parallel and then launch the binary using the correct mpirun command. See also the sample run scripts and the Examples document.

The $AMSBIN/setenv.sh and $AMSBIN/start scripts take care of setting up the environment and starting the binaries. If necessary, it parses the information provided by a batch system and sets up a machinefile (or an appfile) and runs tasks in parallel. Edit the file if you need to customize the way MPI programs are started.

11.2 The atomicdata directory

The directory atomicdata/ contains a large amount of data needed by programs from the AMS package at run time. For example, it contains basis sets for all atoms. Generally you should **not** modify any of the files in this directory.

The basis sets are described in detail in the ADF manual and BAND manual

11.3 The data directory

The directory data/ contains files needed for running the programs and GUI from the AMS package. No modifications should be made to the files in this directory.

11.4 The examples directory

The directory examples/ contains sample script and output files. The files with extension .run are shell scripts that also contain embedded input. Thus, these files should be executed, and not given as input to AMS.

The example calculations are documented in the ADF Examples documentation, and BAND Examples documentation.

11.5 The src directory

The source files are only visible if you have a copy of the source code distribution. The source code files found in the program and library directories and subdirectories thereof have extensions .f, .f90, .c or .cu and contain FORTAN or C/CUDA code. Other source files are include files (files with extension .fh or .h). When compiling, the object files are generated in the folder $AMSHOME/build, and archived into libraries.

Compilation is done by $AMSBIN/foray, and the configure options are located in $AMSHOME/Install/-machinetype-/Forayflags.

The Install directory contains a configure script, some data files which provide generic input for configure (start, starttcl, and some more), a portable preprocessor cpp (based on Mouse cpp) and machine-specific files that are unpacked into the bin folder (precompiled packages), or the build folder (precompiled libraries). The machine-specific folders start with x86 or x86_64.

11.6 The Utils directory

The Utils/ directory contains the “run_test” script and two “rc” scripts. The run_test script can be used to run examples from the examples/ folder and validate their output. The amsrc.sh and amsrc.csh can be used instead of the amsbashrc.sh script in case one uses a C/TC/Z-shell (amsrc.csh), or if bash needs to be avoided (amsrc.sh).

11.7 The Doc directory

All the user documentation for AMS is present in html format in $AMSBIN/Doc. Documentation is also available on the SCM website (https://www.scm.com/support)

11.8 The scripting directory

This directory contains some useful scripts that are part of the AMS package.

CHAPTER

TWELVE

APPENDIX C. DEBUGGING MPI PROBLEMS

The Amsterdam Modelling Suite (AMS) is designed to work out of the box on as many platforms as possible, but it could be that you are experiencing problems with running our programs in parallel. This mostly happens on linux cluster environments, but the information below could also be useful for Windows or MacOS users. However, anything that mentions cluster environment or batch system is linux specific!

12.1 Technical introduction into AMS

Most of our compute programs use Message Passing Interface (MPI) communication for parallel operation. To make AMS as easy to use as possible we ship the MPI runtime libraries within the AMS package, and take care of launching of the MPI programs in parallel in our start script. Because of this setup it is not needed to "mpirun" our programs yourself, so **never do something like "mpirun $AMSBIN/ams"** !

12.1.1 Execution order

When starting one of our programs, for example *$AMSBIN/ams*, you actually call our start script via a symbolic link. This start script then:

- checks which program it should launch, based on the name it was called from or the value of the -x flag
- detects how many cores should be used: -n or the NSCM environment variable are read, when both are empty we detect the cpu core count (see below for cluster environments!)
- sources the *$AMSBIN/setenv.sh* script to set up environment variables
- Write the given input to a temporary file
- detect if and which batch system it is running on
- do the correct parallel MPI launch of the program

`Old pre-AMS2020 info:` When running *$AMSBIN/adf*, this sequence is preceded by a program that first sets up and runs the "atomic create runs" in serial.

12.1.2 The $AMSBIN/start script

As described above, the start script takes care of selecting the number of cores to use for the calculation, sourcing the setenv.sh script, and launching the selected program in parallel. You should not need to edit it if your AMS program fails to start, or does not run on the correct number of cores. In such cases, set the **SCM_MPIRUN_EXE** and **SCM_MPIRUN_OPTIONS** variables in your environment to override the autodetect mechanism in the start script!

The start script tries to detect if it is running on a cluster environment with a batch system such as PBS/Torque, SLURM, SGE or LSB. If it detects one of the environment variables associated with these batch systems, it will leave figuring out the MPI job configuration (cores and nodes) to the MPI library. The MPI library should get the number of cores to run on, and which compute nodes to use (in case of a multi-node calculation) from the batch system. the **NSCM** environment variable and the **-n** flag are ignored in this case!

If no batch system is detected, and both **NSCM** and **-n** are not used to explicitly set the number of cores to use for running the AMS program, then the start script attempts to detect the number of cores on the machine, and uses all available cores. We only count real CPU cores (no hyperthreading or modules with shared FPUs such as AMD Bulldozer), because the AMS programs do not benefit from hyperthreading.

Finally the start script launches the AMS program using the correct mpi commands for the situation.

12.1.3 The $AMSBIN/setenv.sh script

The *$AMSBIN/setenv.sh* script takes care of setting up environment variables needed for running the AMS programs. This includes among others:

- the LD_LIBRARY_PATH variable with locations to the required libraries (and DYLD_LIBRARY_PATH)
- the PYTHONPATH variable

The setenv script also responds to a couple of environment variables, such as **SCM_USE_LOCAL_IMPI**, **SCM_USE_IMPI_2021**, and **SCM_USE_LOCAL_OMPI**. See Appendix A on Environment Variables for details.

12.1.4 MPI runtimes

The Amsterdam Modelling Suite ships with the required MPI runtime inside the package. For MacOS this is OpenMPI, for Windows this is MSMPI, and for Linux both IntelMPI and OpenMPI flavors exist. When working on Linux we advise to use the IntelMPI library, as this provides the best out-of-the-box experience. The OpenMPI runtime has no batch system integration build into it, so it will most likely not work under a batch system. The runtimes can be found in *$AMSBIN/IntelMPI* or *$AMSBIN/openmpi*, and are loaded into the environment by the setenv.sh script. See the Additional information section for more details.

12.2 Debugging MPI issues

For non-batch system (desktop/laptop usage) related MPI issues, please contact support@scm.com as these should almost never happen. However, linux users might want to try using a newer IntelMPI runtime by setting **export SCM_USE_IMPI_2021=1** in their environment.

For cluster usage we always advise to use the IntelMPI version. AMS ships with the IntelMPI runtime inside the package, you do not need to install this separately. IntelMPI can be used on Cray systems as well via the ABI compatibility, see the Additional information section for more details. If you for whatever reason need to use OpenMPI instead, make sure to compile and use your own OpenMPI version that matches the version used to build AMS (currently 4.1.1), and set SCM_USE_LOCAL_OMPI=true in the environment. Make sure to to read the Additional information and known issues document before you dive into debugging MPI! If you know how to launch MPI programs on your cluster and the autodetect mechanism is not giving the correct results, you can set the **SCM_MPIRUN_EXE** and

SCM_MPIRUN_OPTIONS variables in your environment directly (make sure to "export" them). The start script will then use those variables to launch the mpi job. See Appendix A on Environment Variables for details.

The generic scheme for debugging IntelMPI / batch system issues consists of the following steps:

1. Start by setting up the IntelMPI runtime libaries in the environment: load it from a module, or load your AMS environment and execute ". $AMSBIN/setenv.sh" without the quotes. Keep in mind that in order to test the IntelMPI 2021 runtime shipped with AMS2021, you will need to execute "export SCM_USE_IMPI_2021=1" before sourcing the setenv.sh script.
2. get the intel mpirun to work. Environment variables that are useful for this are I_MPI_DEBUG=100 (or higher), I_MPI_OFI_PROVIDER_DUMP=1 (for information about the interconnects detected by IntelMPI), the I_MPI_FABRICS (for selecting a specific fabric, options depend on the IntelMPI runtime version) and I_MPI_HYDRA_TOPOLIB. See the IntelMPI 2018u3 developer reference manual (https://software.intel.com/en-us/download/intel-mpi-library-for-linux-os-developer-reference-2018-update-3) for details on these variables for the default IntelMPI runtime. If you are using the IntelMPI 2021 runtime, see IntelMPI current developer reference manual (https://software.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-linux/top/environment-variable-reference.html)
3. get a small mpi test program that's compiled with IntelMPI and a Fortran compiler. To make things a bit easier you can download an example with both binary and source code here (https://downloads.scm.com/distr/MPItest.tgz).
4. try to mpirun the test program (mpirun ./mpitest) on a single node via the batch system. This usually respects the batch system reservation. In case it doesn't you will need to dive into the manuals to see what kind of environment variables you need to set, or arguments you need to give to get it to integrate with the batch system. IntelMPI developer references and guides (https://software.intel.com/en-us/articles/intel-mpi-library-documentation-overview) help out with this, but make sure to get the manual matching your IntelMPI version! The manual of your batch system might also hold some clues.
5. once single node jobs work, then it is time to see what needs to be done for multi-node jobs. Usually this can be set up via integration with the batch system, but if that fails you could always write a bit of code that generates a "machine file", and tell IntelMPI to use this file with the -machinefile flag (see this (https://software.intel.com/en-us/mpi-developer-guide-linux-controlling-process-placement) for how a machine file should look)

It might be that the default IntelMPI runtime shipped with AMS (currently 2018 update 3) needs to be updated for a better integration with your batch system. In such a case you can try the IntelMPI 2021 runtime (also shipped with AMS) by setting "export SCM_USE_IMPI_2021=1" in your environment. Or you can download and install a newer IntelMPI runtime package (those are free to use), and repeat the previous steps. If the downloaded IntelMPI works better than the 2018 or 2021 runtime shipped with AMS, then you can set the "export SCM_USE_LOCAL_IMPI=true" environment variable. This tells AMS not to load the distributed IntelMPI into the environment, but instead use the one already available. (You do not need to recompile AMS for this!)

Finally: when requesting support for MPI issues via support@scm.com, make sure to include as much information as possible. For example: which batch system are you using, what version is it, do you have a special interconnect such as Infiniband, and of course always send us all the input and output files produced by the failed job, including the sdtout and stderr of the batch system. From our experience the fastest way to resolve such issues is if somebody from SCM can work directly on the machine. Therefore you might want to consider setting up a form of temporary remote access for an SCM employee to help you out.

12.3 IntelMPI on Linux Quirks

AMS with IntelMPI on Linux uses the 2018u3 runtime by default, because in our experience this is still the fastest and most reliable IntelMPI runtime. However, for newer CPUs (both AMD and Intel Xeon), the 2018 runtime sometimes has startup problems. In such cases we suggest trying the IntelMPI 2021.1 runtime, which we also distribute with AMS, by setting "export SCM_USE_IMPI_2021=1" in your environment. If this does not work for your system, you can download and install a newer (or older) IntelMPI runtime yourself, and use this by setting "export SCM_USE_LOCAL_IMPI=1" in your environment.

The IntelMPI 2021.2 and 2021.4 runtimes have been found to deadlock for some operations on AMD Zen3 processors. They do seem to work fine on Intel CPUs. The IntelMPI 2021.6 runtime seems to work fine for both Snellius (the new Dutch supercomputer at SARA), and works without problems on our Zen3 systems, as well as Intel-based CPUs.

CHAPTER

THIRTEEN

APPENDIX D: ADVANCED PACKAGE MANAGER USAGE

13.1 Overview

AMSpackages helps you maintain optional components of the Amsterdam Modeling Suite. Features installed through AMSpackages integrate with both the GUI and command line programs, and can be managed through either interface. All package installs require a valid license.

Note: AMSpackages behavior changed significantly in AMS2026. This page describes the package manager for AMS2026 and later. If you are using AMS2025 or an older release, see the documentation for previous AMS versions (https://www.scm.com/downloads/documentation-previous-versions/).

New in version AMS2026.101:

- The default behavior of AMSpackages now installs packages for all users.
- The default location for shared packages is now inside `AMSHOME` on all platforms.
- A `--single-user`/`-u` flag was added to fall back to the old per-user behavior.

13.2 Managing packages from the command line

On the command line, users can use `"${AMSBIN}/amspackages"` to access the package manager interface. There are several subcommands that can be used to access several features such as installing, updating, or removing packages. Below the most commonly used options are explained.

To see a full list of options you can use `"${AMSBIN}/amspackages" --help`. Any of the subcommands may also have specific options, which can be seen when using `--help` behind the subcommand. For instance, for `"${AMSBIN}/amspackages" install --help`.

13.3 Helpful commands and global options

Besides the main install, update, and remove operations, the following commands are often useful when working with AMSpackages from the command line:

- `"${AMSBIN}/amspackages" list` shows available and installed packages
- `"${AMSBIN}/amspackages" check lfdft` checks whether a package is installed
- `"${AMSBIN}/amspackages" loc lfdft` prints the install location of a package

- `"${AMSBIN}/amspackages" clean` removes cached downloads and leftovers from aborted installations
- `"${AMSBIN}/amspackages" debug` prints debugging information in markdown format
- `"${AMSBIN}/amspackages" env` shows the environment variables exported by AMSpackages

Useful global options include:

- `-v` or `--verbose` for more detailed output. You can provide this option multiple times.
- `--repository` to use a specific online or local repository definition file
- `--timeout` to change the network timeout for downloads
- `--no-cache` to skip cached repository data
- `--no-shared` to ignore packages installed in the shared location

For example:

```
"${AMSBIN}/amspackages" -v list --format wide
"${AMSBIN}/amspackages" --repository /path/to/AMS2026.1.yml list
"${AMSBIN}/amspackages" --no-cache check lfdft
```

13.4 Finding available packages

New in version AMS2024.101.

- The `display` command was renamed to `list`.
- `yaml` was added as an available format.

The `list` command will show available packages.

If you wish to refer to a package, you can use the `ID` to install, remove it, or check if it is installed. By default, list provides a long list of packages, but a wide view is also available.

For example:

```
> "${AMSBIN}/amspackages" list
AMSpackages Overview
Date: Tue May 11 12:07:01 2021
Repository: https://downloads.scm.com/Downloads/packages/
Package directory: /home/user/.scm/packages/AMS2021.1.packages
Name                                                [id]|     Instl. Version     |  ␣
↪Disk usage   |     Avail. Version     | Licensed? | Req. licenses
=====================================================|=========================|================|==
ADFCRS-2018 Database                              [adfcrs]|                         |   ␣
↪             |     2018: build 1      |    YES    | CRS 2021.1
LFDFT atomic database                               [lfdft]|      1.0: build 0      |   ␣
↪1.42 GiB     |      1.0: build 0      |    YES    | ADF 2021.1
All ML Potential backends                        [mlpotentials]|                         |   ␣
↪             |     1.0.0: build 0     |    YES    | MLPOT 2021.1
PiNN ML backend                                    [pinn]|                         |   ␣
↪             |     0.3.1: build 0     |    YES    | MLPOT 2021.1
Quantum ESPRESSO                                    [qe]|                         |   ␣
↪             |      6.3: build 1      |    YES    |
SchNetPack ML backend                            [schnetpack]|                         |   ␣
↪             |     0.3.1: build 0     |    YES    | MLPOT 2021.1
sGDML ML backend                                  [sgdml]|                         |   ␣
```

(continues on next page)

(continued from previous page)

```
↪            |     0.4.4: build 0     |    YES    | MLPOT 2021.1
TorchANI ML backend                          [torchani]|                          |  ␣
↪            |     2.2.0: build 0     |    YES    | MLPOT 2021.1
```

Alternatively, machine parsable output can be obtained using `--format=yaml`.

The `list` command also supports `--format=long` and `--format=yaml`. The wide format is convenient for terminal use, while yaml is useful in scripts and automated checks.

Table 13.1: Available AMSpackages

Package name	Package ID	Available version
ADFCRS-2018 Database	adfcrs	2018: build 5
AIMNet2 Environment (CPU Only)	aimnet2-cpu	1.0.0: build 224
AIMNet2 Environment (CUDA 12.8)	aimnet2-cu128	1.0.0: build 224
Amsterdam Modeling PySuite Environment	amsterdam-modeling-pysuite	1.0: build 224
FAIRChem Environment (CPU Only)	fairchem-cpu	1.0.0: build 224
FAIRChem Environment (CUDA 12.8)	fairchem-cu128	1.0.0: build 224
Infretis Environment	infretis	1.0.0: build 224
LFDFT atomic database	lfdft	1.0: build 0
M3GNet Environment	m3gnet	1.0.0: build 224
MACE Environment (CPU Only)	mace-cpu	1.0.0: build 224
MACE Environment (CUDA 12.8)	mace-cu128	1.0.0: build 224
Subgraph Sigma Profile Estimation (SG1) Database	molsg_sg1db	1.0: build 3
NequIP Environment (CPU Only)	nequip-cpu	1.0.0: build 224
NequIP Environment (CUDA 12.8)	nequip-cu128	1.0.0: build 224
OLED material database	oledmatdb	2023.1: build 2
Quantum ESPRESSO (AMSPIPE)	qe	7.1: build 206
TorchANI Environment (CPU Only)	torchani-cpu	1.0.0: build 224
TorchANI Environment (CUDA 12.8)	torchani-cu128	1.0.0: build 224
Zacros-post	zacros_post	1.02: build 2

13.5 Installing a package

To install a package, use the `install` command and specify the ID of the package.

```
"${AMSBIN}/amspackages" install lfdft
```

Multiple packages can be specified on the same line for installation. If you wish to install all available packages, *all* can be used as a shortcut.

```
"${AMSBIN}/amspackages" install all
```

By default, AMSpackages installs packages in shared/admin mode inside `AMSHOME`. If `AMSHOME` is not writable, it automatically falls back to single-user mode. To force single-user installation, use `-u` or `--single-user`:

```
"${AMSBIN}/amspackages" --single-user install lfdft
```

Some packages such as the Machine Learning potentials are installed into your amspython environment. Other packages are installed in the shared package location inside `AMSHOME` by default.

In single-user mode, the default package locations are:

- for Linux users: `$HOME/.scm/packages`
- for MacOS users: `$HOME/Library/Application Support/SCM/packages`
- for Windows users: `%LOCALAPPDATA%/SCM/packages`

13.6 Installing a package from a script

If you want to install a package non-interactively, for instance inside of a batch script, use the `--yes` option after the install command. This will answer yes to any interactive prompts.

```
"${AMSBIN}/amspackages" install --yes lfdft
```

As an alternative you can set an environment variable. Add `export SCM_AMSPKGS_NONINTERACTIVE=True` to your script to answer yes to all prompts.

Other useful install options are:

- `--force-reinstall` to reinstall an already installed package
- `--downgrade` to allow installation of an older available version
- `--force` to install despite user modifications in the target package directory

13.7 Check if a package is installed

If you want to make sure that a package is installed, you can use the `check` command.

```
"${AMSBIN}/amspackages" check lfdft
```

The exit status will be 0 if a package is installed, and non-zero otherwise. For detailed information, you can pass the verbose (`-v`) flag. This will tell you what version is installed.

```
> amspackages -v check lfdft
04-20 17:26:28 lfdft is installed: Ligand Field DFT v[1.0] - build:0␣
↪[974661607e9f46c78b6d875e12edfb7a]
```

Additional check options are available for stricter verification:

- `--modifications` fails if the package was changed externally
- `--licenses` also validates license availability and version compatibility
- `--pip` checks for missing Python packages using pip

For example:

```
"${AMSBIN}/amspackages" check --licenses --modifications lfdft
```

13.8 Removing a package

Removing a package is done by using the `remove` command. This command can take a list of packages as arguments. For example, to remove the Ligand Field DFT package use:

```
"${AMSBIN}/amspackages" remove lfdft
```

While we advise against it, there may be times when a user has accidentally or intentionally made changes or saved new files inside the directories of one of the packages. When trying to remove such a package, to prevent data loss you will be warned that a package is modified. In order to remove packages despite modifications, use the `--force` flag.

```
"${AMSBIN}/amspackages" remove --force lfdft
```

13.9 Updating a package

At times a newer version or build of a package may be available. To install an update, use the `update` command.

```
"${AMSBIN}/amspackages" update lfdft
```

You can also run `update` without package arguments to update all installed packages.

Note: The update mechanism usually removes user modifications. Therefore just like the remove command, you will have to specify the `--force` flag to update such packages.

If you install a new version of AMS, reinstall any optional packages in that new installation instead of copying them from an older AMS version.

Other useful update options are:

- `--fix` or `--fix-unsupported` to replace unsupported package versions, for example after manual pip changes
- `--downgrade` to allow updating to an older available version when needed
- `--yes` for non-interactive updates

13.10 Locating an installed package

If you need the exact installation path of a package, use the `loc` command.

```
"${AMSBIN}/amspackages" loc lfdft
```

This is useful for troubleshooting, verifying where a package ended up, or locating package files manually.

13.11 Cleaning cached files

AMSpackages can keep downloaded archives, uv cache data, and partially installed package directories. To reclaim disk space or clean up after interrupted operations, use:

```
"${AMSBIN}/amspackages" clean
```

You can keep selected caches by using:

- `--no-downloads` to keep downloaded package files
- `--no-uv-cache` to keep the uv cache
- `--no-installs` to avoid cleaning package folders left by aborted installs

13.12 Debugging and environment inspection

For support and troubleshooting, the following commands are useful:

- `"${AMSBIN}/amspackages" debug` prints a markdown report with debugging information
- `"${AMSBIN}/amspackages" debug --no-log` omits the appended log file
- `"${AMSBIN}/amspackages" env` shows the environment variables exported in the pkgrc file

These commands are especially useful when preparing information for SCM support or when diagnosing configuration issues in shared installations.

13.13 Using a local mirror with AMSpackages

If you need to create or update the local mirror itself, see *Appendix E: Mirroring the SCM package repository* (page 91). This section describes only how to point AMSpackages to a mirror that already exists.

Recommended:

Place your downloaded copy of the AMS repository in AMSHOME. If the root file is located at: `${AMSHOME}/repository/AMS2026.yml`, amspackages will automatically recognize and use it. If you need to use an an alternative location, on the command line you can point the package manager to the downloaded repository as follows

```
amspackages --repository "/full/path/to/Downloads/packages/AMS2026.1.yml" list
```

Or, you can set the `SCM_AMSPKGS_REPO` environment variable in your shell, or in the *configuration file* (page 87).

This will allow other programs that use the package manager to use the same repository for checking and installing packages.

In your shell (e.g. in *.bashrc*):

```
export SCM_AMSPKGS_REPO="/full/path/to/Downloads/packages/AMS2026.1.yml"
```

In the configuration file:

```
SCM_AMSPKGS_REPO: /full/path/to/Downloads/packages/AMS2026.1.yml
```

13.14 Instructions for administrators

It is typical for several users to access and use the same installation of AMS. On shared systems, such as a compute cluster, it is often desirable to share installed optional components. This prevents redundant installations of the same feature. On shared installations, the default behavior is already to install shared packages inside `AMSHOME` when that location is writable. Users who cannot write to `AMSHOME` can use single-user mode instead. If you want to force shared/admin mode explicitly, use the `--admin` flag. For instance, to install a shared package explicitly, use

```
"${AMSBIN}/amspackages" --admin install lfdft
```

Instead of in the user directory, packages will be installed in the shared installation. Users need read-only access to the shared installation directory.

AMSpackages supports environment variables that can override default options. The shared package location can be overridden using `SCM_AMSPKGS_SHAREDDIR`. The single-user package location can be overridden using `SCM_AMSPKGS_USERDIR`. You should add any shared-location override to the persistent configurations, so users can find the installation path that you used for installing.

13.15 Installing python packages for users

Warning: SCM does not provide technical support for packages from pypi.org or other sources. Do not install python packages from untrusted developers.

Python packages may be pre-installed into amspython by admins. The simplest method is to create the folder "python" in AMSHOME, the root of the AMS installation.

After that, you must open a new shell before installing packages.

```
"${AMSBIN}/amspython" -m pip install lxml # installs the lxml package from pypi.org
```

When using the paths above, the venv containing all python packages will be located in

```
$AMSHOME/python
```

Alternatively, one can export `SCM_PYTHONDIR` to a directory of their choosing. Be sure to also export this variable at run time / for all users or the installed packages may not be detected.

13.16 Persistent configurations

Configuration file

YAML syntax

The configuration file uses YAML syntax. Environment variables can be specified using the variable name, followed by `:`, a space, and then the value.

The `amspackages` command has a lot of different options that users or admins may need to change to suit their particular system. The full list of configuration options can always be seen when running `${AMSBIN}/amspackages --help`.

The usage text contains the name of the environment variables that AMSpackages looks for between `[]` brackets. A user can export these in their shell environment, or an administrator can add them to the configuration file to make them persistent throughout the usage of the package manager. These options will also affect the operation of the package manager gui. The configuration file will make settings persistent for all users of the installation of AMS. The file is located inside AMSBIN, under

```
"${AMSBIN}/amspackageslib/config.yml"
```

You can open this file in a plain text editor to make changes. Inside you can find several variables that are set by default. Please take care not to accidentally remove the pre-existing variables, as they are necessary for the package manager to function. You can add your own variables to the file, and edit variables such as the repository, for instance to use a *local copy* (page 33).

Some example variables you may wish to set

```
SCM_AMSPKGS_REPO: /downloads/SCM/packages/AMS2025.1.yml # A local copy of the package␣
↪repository
SCM_AMSPKGS_SHAREDDIR: /shared/SCM/packages/ # Admin provided packages will be␣
↪installed here
SCM_AMSPKGS_USERDIR: ~/.scm/packages # User packages will be installed here.
```

Note that using variables (for example `$HOME`) inside the names of paths is not supported, but you can use a ~ as a shortcut for home in path names. If you require a different configuration per user, it is recommended to set an environment variable inside the user environment instead of using the configuration file.

Note: Variables not prefixed with "SCM_AMSPKGS", for example **SCM_PYTHONDIR** can not be set in the config.yml file. They should always be set and exported in the shell environment or they might not take effect.

13.17 Environment variables

AMSpackages can be configured through the following environment variables.

SCM_AMSPKGS_CONFIG: If set, variables will be read from this file instead of the default configuration file.

SCM_AMSPKGS_REPO: The location of the repository root definition file. Can be either a URL or a path to a local yaml file.

SCM_AMSPKGS_USERDIR: The location for single-user package directories.

SCM_AMSPKGS_SHAREDDIR: The location for shared/admin package directories.

SCM_AMSPKGS_VERBOSITY: Increase output verbosity (integer value between 0-4).

SCM_AMSPKGS_SSL: Set this variable to `False` to disable SSL, or point it to a path containing certificates.

SCM_AMSPKGS_NONINTERACTIVE: Set this variable to `True` to run without user input (e.g. yes to all prompts).

SCM_AMSPKGS_ADMIN: Set this variable to `True` to force shared/admin mode.

SCM_AMSPKGS_NOSHARED: Set this variable to `True` to ignore any packages installed in the shared installation..

SCM_AMSPKGS_NOCACHE: Set this variable to `True` to ignore cached repository information.

SCM_AMSPKGS_PIPSTRICT: Set this variable to `True` to always strictly match exact package versions for python packages.

SCM_AMSPKGS_LIST_FORMAT: Set to `wide`, `long`, or `yaml` for default display mode in the `amspackages list` output.

CHAPTER

FOURTEEN

APPENDIX E: MIRRORING THE SCM PACKAGE REPOSITORY

14.1 Overview

A local package mirror is a copy of the SCM package repository that you keep on your own machine, shared filesystem, or internal network. This is useful when target systems cannot access the internet directly, when compute nodes are behind a firewall, or when you want to prepare one repository copy and reuse it across multiple AMS installations.

This appendix describes how to create a local copy of the SCM package repository. For information on telling AMSpackages to use that mirror, see *Appendix D: Advanced package manager usage* (page 81).

> **Warning:** Never install packages from an untrusted source. Only use a mirror copied directly from SCM.

14.2 Using `amsrepomirror`

The preferred way to prepare and update a local SCM package mirror is with `amsrepomirror`. This tool is distributed separately from AMS and is intended to simplify repository mirroring compared with the manual methods below.

Detailed usage documentation for `amsrepomirror` will be added here in a future update. Until then, you can continue to create mirrors manually using the workflows below.

14.3 Manual mirroring methods

If you are not using `amsrepomirror`, the repository can still be mirrored manually. You can find a listing of the repository contents on our website (https://downloads.scm.com/Downloads/packages/listings/).

Use the listing that matches your version of AMS, or you may have unexpected issues with installed packages. You will need to download these files and preserve their directory structure.

14.4 MacOS and Linux

14.4.1 Method 1: wget

Note: On MacOS, you will need to install wget first. You can use brew (https://brew.sh/), or macports (https://www.macports.org) to install it.

On Unix systems, wget can be used to achieve this from the command line. To download the repository for AMS2025.1 you would use the following from the command line:

```
wget --user scmuser --password scmpasswd -xi https://downloads.scm.com/Downloads/
↪packages/listings/AMS2026.1.txt
```

This will create the `downloads.scm.com` folder in your current working directory, containing just the path to the repository.

14.4.2 Method 2: python

You can download the following `python script` to download a copy of the repository. If you have a local installation of AMS, then you can run it with amspython. The script can resume after interruption, and update files that have changed in size.

It uses the following command line options

```
usage: download_repository.py [-h] [--strict-authentication] [--no-ssl] [--local-
↪platform-only] [--delete-unlisted] LISTING_URL DOWNLOAD_FOLDER USER PASSWORD

positional arguments:
  LISTING_URL           Url pointing to the file listings for the repository.
  DOWNLOAD_FOLDER       A local directory for storing your download.
  USER                  Username for downloading from the website.
  PASSWORD              Password for downloading from the website.

optional arguments:
  -h, --help            show this help message and exit
  --strict-authentication
                        If supplied this script will exit on 401 errors. By default,␣
↪this script will skip files on 401 errors.
  --no-ssl              Don't use SSL verification.
  --local-platform-only
                        Skip downloading files for platforms other than the local one␣
↪(e.g. skip MacOS & Linux packages on Windows).
  --delete-unlisted     Delete files that are not present in the listing from the␣
↪destination (e.g. clean up old versions of packages).

All positional arguments are REQUIRED.
```

For example, you can run it as

```
amspython download_repository.py https://downloads.scm.com/Downloads/packages/
↪listings/AMS2026.1.txt ~/Downloads/my/repo scmuser scmpassword
```

If you want to use your own version of python, you need to install tqdm, and requests. Note that the script requires python 3.6 or greater.

```
python -m pip install tqdm requests
python download_repository.py --help
```

14.5 Windows

For Windows users, we provide this `powershell script`.

Download this script, along with the listing file. You can right-click the script file in the file browser and select `Run with Powershell` to start it.

It will open a blue window displaying the progress, as well as some interactive prompts that will require you to provide the listing file, the location to store the download, and your SCM user account and password.

Warning: Don't close the blue powershell window while the program is running.

The script will automatically start downloading the files to the folder of your choosing. It may take a while to download all the files. When it is done, the blue window will display a message saying that it is safe to close.

Note: Your computer may not allow you to run powershell scripts by default. If you receive a warning about not being allowed to run scripts, first open a open a `powershell` window using `win + R` on your keyboard, and typing powershell. In the powershell window that opens type `powershell -ep Bypass C:\Path\To\Script`. Fill in the correct path to the script you downloaded.

Below, you can see a demonstration of how you can download and use the local copy on Windows.

You will need an internet connection to see the video.

14.6 Using the mirror with AMSpackages

After creating a mirror, point AMSpackages to it as described in *Appendix D: Advanced package manager usage* (page 81). In many common cases, AMSpackages will also detect a local mirror automatically when a folder named `repository` is placed inside `AMSHOME`, and it contains the matching `AMS2026.1.yml` file.

CHAPTER

FIFTEEN

COMPILING AMS FROM SOURCES

THIS INFO IS ONLY RELEVANT FOR PEOPLE WITH A SOURCE CODE LICENSE! If you are unsure what this means, you can most likely skip reading this page!

Compiling AMS from sources by end users is not supported on Windows and macOS. The following instructions apply to Linux only. Compiling AMS2025 from sources is supported using Intel OneAPI 2024.1.0, including ifort, MKL and IntelMPI on linux. For more details on the recommended compilers and libraries, see the *Platform Specific Information* (page 97).

15.1 Unpacking the distribution

Installing AMS with recompilation in mind is somewhat different from the binary-only installation. The downloaded source and binary tarballs must be unpacked in the following order (using IntelMPI on x86_64-linux in this example):

```
# First sources
tar xzf ams2025.102.src.tgz
# Then binaries
tar xzf ams2025.102.pc64_linux.intelmpi.bin.tgz
```

The result will be a ams2025.102 directory containing both binaries (for your platform) and sources.

15.2 Setting up environment

This document assumes you are using a bash shell. If you are using a different shell, some commands might need to be modified. The following environment variables should be set:

- I_MPI_ROOT: this variable must be set if compiling with IntelMPI on linux (the default)
- MPIDIR: may be needed in the case of compiling AMS with non-default MPI, for example OpenMPI on linux.
- MATHDIR/MKLROOT: this should be set to the the MKL root folder. If MKLROOT is defined and MATHDIR is not, then MKLROOT will be used.
- AMSHOME/AMSBIN/AMSRESOURCES/…: the standard environment variables discussed in the Installation manual, also needed for running AMS. The easiest way to set them is to source the amsbashrc.sh script:

```
cd ams2025.102
. amsbashrc.sh
```

15.3 Running Install/configure

After unpacking everything and setting up your environment properly, you need to run the *configure* script. This script is located in the $AMSHOME/Install directory, and it must be executed from the $AMSHOME directory. The script replaces some files in the bin directory with versions specifically targeted for your system. Further, *configure* creates the buildinfo file that you will need to compile AMS.

To see what options the configure script supports, use configure -h:

Example:

```
cd $AMSHOME
Install/configure -h
```

Configure can set up multiple build targets with different configuration options using the -b flag. The options regarding your build target should be wrapped in quotes following a -b flag, starting with the build name. The -b flag can be used multiple times to create different build targets. For example, to create a target with all current release build options:

```
Install/configure -b "release -p intelmpi -dynamicmkl -plumed"
```

If a different MPI version is needed (for example OpenMPI) it can be selected with the -p flag:

```
Install/configure -b "mydefaulttarget" -b "myompitarget -p openmpi"
```

15.4 Compiling AMS

Next, you need to compile the sources by executing foray located in $AMSBIN. Foray supports parallel compilation to make things faster, use -j N to tell foray you want it to use N processes (for example: -j 4 on a quadcore machine):

```
cd $AMSHOME
bin/foray -j 4
```

After a while, depending on the speed of your computer, everything should be ready to use, just as if you had installed the precompiled executables but now with your modifications included. Use *bin/foray -h* to get a list of all foray options.

CHAPTER

SIXTEEN

PLATFORM SPECIFIC INFORMATION

The following table displays the platforms, compilers and MPI versions that we have used to prepare the executables and object libraries. Other versions (especially older ones) are not guaranteed to work with this release of the Amsterdam Modeling Suite 2025. Please contact us if you have problems running AMS on your platform.

16.1 Machine-specific comments

- **64-bit Windows**: The Windows version has been prepared under Windows 11 and tested under Windows 10 and 11. This version has not been tested on other versions of Windows but it might also work on Windows 7. AMS2025 is not compatible with Windows XP, and only works on 64-bit installations of Windows. Please read Installation instructions for the list of known issues and important information about installing and configuring the package on Windows.
- **x86-64 Linux**: The x86-64 Linux version requires both a 64-bit processor and a 64-bit operating system. The processor must support AVX2 instructions. The minimum supported GLIBC version for AMS2025 is 2.17 for both the GUI and the compute engines. The users with an earlier GLIBC version should upgrade their system. If your OS is older than CentOS 7 / RHEL 7, you might also experience problems.
- **Cray XT, XE, and XC**: The Linux IntelMPI distribution of AMS2025 is compatible with cray machines using the MPICH ABI compatibility.
- **AMD-Zen version**: The AMS2025 Optimized for AMD-Zen version is targeted at AMD Zen, Zen2, Zen3 and Zen4 processors with AVX2 instructions. This currently includes all AMD Ryzen, Threadripper and Epyc processors. It is built using AMD Optimizing CPU Libraries (AOCL).

Table 16.1: Platform Specific Information

Platform	Operating System	Fortran compiler	C Compiler	MPI brand & version	Math Library
Windows	Windows 10 64-bit, MinGW msys2-20161025	Intel Fortran, 2021.12.0	Microsoft Visual C++ 19.29.30130.2	Microsoft MPI 10.1.12498.18	Intel MKL 2021.4.0
Linux	CentOS Linux 7.9, glibc 2.17	Intel Fortran, 2021.12.0	GCC 10.3.0	IntelMPI 2018 update 3[1]	Intel MKL 2024.1.0
Linux with OpenMPI	CentOS Linux 7.9 glibc 2.17	Intel Fortran, 2021.12.0	GCC 10.3.0	OpenMPI 4.1.1	Intel MKL 2024.1.0
Linux for AMD	CentOS Linux 7.9 glibc 2.17	Intel Fortran, 2021.12.0	GCC 10.3.0	IntelMPI 2018 update 3[Page 98, 1]	AMD AOCL 4.0
Arm64 Mac OS	macOS 13.2	Gfortran 12.2.0	Apple LLVM version 14.0.0	OpenMPI 4.1.4	Apple Accelerate Framework

[1] The Linux IntelMPI variants also ship with an IntelMPI 2021.1 runtime, use “export SCM_USE_IMPI_2021=1” to select it

CHAPTER

SEVENTEEN

FAQ

17.1 Can the GUI in AMS2020 read older inputs (.adf) and results (.t21, .*kf)?

The GUI will try to convert old .adf files to new .ams files as best as it can.

We attempt to keep the 2020 GUI version as much backwards compatible as possible for visualizing results such as spectra, orbitals, band structures, trajectories etc. from older .t21 and .*kf files.

Please let us know at support@scm.com with your old input / result files if it doesn't work for you.

17.2 Which hardware should I get?

In general, the best hardware will be: the latest generation of processors, 4GB RAM per core and an SSD. See also the summary of a hardware FAQ (https://www.scm.com/news/hardware-questions-and-answers-recommendations-gui-best-value) session from October 2020. For the latest information, see this *installation manual* (page 9).

Based on your budget, an older processor, 2GB of RAM per core and a standard hard drive may also be sufficient for running ADF jobs. Since floating-point performance is crucial for ADF, we do not recommend to use hyperthreading or use all Opteron logical cores (i.e. run one process per floating point unit). Note that since chips are continuously improved, specific recommendations may change. As of mid-2017, these configurations should give the best performance for ADF:

Desktop or laptop: an Intel i7 processor from the latest generation, 16GB of RAM or more, and an SSD (a fusion drive may also work).

High-end workstation or a cluster: a dual-CPU Xeon SP setup with at least 2GB, but better 4GB, of RAM per core and an SSD. Using an SSD becomes more essential as the number of cores per node increases. For calculations on multiple nodes, get a fast interconnect (Infiniband).

Update mid-2019: Rome, the latest Zen 2 AMD architecture (Epyc, 3rd generation Ryzen), give very good performance for ADF. 4GB RAM per core and SSD are still recommended. Make sure you download the Linux binaries optimized for AMD Zen.

Similar requirements should hold for BAND (maybe you can benefit from even more disk space) and ReaxFF (less memory intensive than ADF).

17.3 I receive this error on my Apple Silicon mac: Illegal Instruction 4

`Illegal Instruction: 4` indicates an illegal CPU instruction. You have probably installed the wrong architecture version of AMS.

The x86_64 intel version of AMS should not be used on Apple Silicon machines (M1, M2, M3), which have arm64 architecture. For macOS users we provide a native version for arm64 architectures that is much faster and should not produce such errors.

Please take the following steps to ensure proper functioning of AMS:

- Delete the old x86 Application (drag it to the bin).
- Delete the folder `/Users/username/Library/Application Support/SCM` which may contain x86 packages from the old installation.
- Install the version labeled Apple Silicon from our downloads page (https://www.scm.com/support/downloads).

17.4 I can not open the ADF / AMS app on MacOS Catalina

If you open AMS for the first time from your applications, you'll see a notice "AMS20xx.xxx" can't be opened because Apple cannot check it for malicious software. To fix this:

- In the Finder on your Mac, locate the app you want to open.
- Control-click the app icon, then choose Open from the shortcut menu.
- Click Open.

The app is saved as an exception, and you can open it in the future by double-clicking it or from your Applications on the Dock.

17.5 I only have OpenGL 1.x on my computer. Can I still use AMS?

Yes! From on AMS2019.304 you can use *software rendering instead* (page 50). To activate the software rendering, type the following command in your command line before starting the GUI:

```
export SCM_OPENGL_SOFTWARE=1
```

To make this permanent, just add it to your $HOME/.bashrc file.

Note: This option replaces the former

```
export SCM_OPENGL1_FALLBACK=1
```

Users with older versions can still make use of the old – less efficient – OpenGL 1.4 fallback option, as described for *AMS2019/18 or ADF2017* (page 49).

Please contact support@scm.com if none of the above solves the issue.

17.6 I have memory issues with my batch system: how do ADF and BAND handle shared memory?

Most programs in AMS are MPI-enabled and will use both private and shared memory.

For clusters with a job scheduler, it may be useful to look into how the batch system *handles shared memory* (page 37). The scheduler could think that the shared memory allocated by ADF or BAND is used per core, resulting in severely overestimated memory usage.

For the SLURM scheduler the system administrator should set JobAcctGatherParams=UsePss so that the shared memory is treated properly. For PBS or Torque schedulers, you should have your system administrator look into this and discuss with support@scm.com the best solution.

17.7 My ADF inputs and scripts no longer work with AMS2018? And AMS2020?

In AMS2020 a major overhaul of the input for ADF has been made, unifying inputs more strongly across modules in the suite.

With the restructuring of our code base and introduction of the AMS driver we have rewritten the input handling to improve cross-engine compatibility.

In most cases, the GUI will have backwards compatibility. And so will amsprep and amsreport scripting tools. However, if you use command line input files and home-grown scripts, you will find that unfortunately these will no longer work with our 2018 version.

17.8 How can I script with python or use command line tools on Windows or MacOS?

Starting with AMS2018, just click help -> Command-line (Windows) or Terminal (MacOS) in any GUI window to start a terminal with all the environment variables set properly.

2017 and earlier versions On Windows, use the adf_command_line.bat tool to open up a command line with the ADF environment set up. You can use 'sh' to run a shell command or by itself to open a rudimentary shell.

On a Mac open a terminal and type a dot. Then open a GUI and go to Help: Show AMSHOME in Finder in the menu, drag that to the terminal and press enter (see also video).

17.9 Will the integrated GUI work as an interface to all my SCM products?

Yes, the integrated GUI works with all our software as well as with MOPAC and Quantum ESPRESSO for which also provide binaries.

17.10 I can't see special characters in the GUI?

The GUI used unicode characters in many places. For example, the proper characters for Angstrom, degrees Celsius, and many more. You can see examples by starting AMSinput, selecting a Geometry Optimization, and then going to the details page (by clicking the … button). There you should see a couple of Angstrom symbols if everything is working as intended.

On most machines this will just work out of the box. If not, the issue might be that the default font on the machine does not support such characters. Try installing a font line DejaVU Sans and Monospace DejaVu. On CentOS you can do this (as root) with the command:

```
yum install dejavu-sans-fonts dejavu-sans-mono-fonts dejavu-serif-fonts
```

17.11 Do I need the source code for the Amsterdam Modeling Suite?

No, usually you don't need the source code. We collaborate with all major hardware vendors to port and optimize our code on the most popular platforms. The binaries work out-of-the-box in parallel with fast interconnects through included MPI libraries or with the native parallel libraries (MPT, POE). The binaries are also shipped with linked-in mkl libraries, and are thoroughly tested against a test set before they are released. Should the latest binary for your system not be available, we are happy to help you port it to your platform. If you are an experienced system administrator you can also compile AMS yourself e.g. on an unusual supercomputing platform. In this case we do not charge you for using the source code.

Only when you want to modify the source code to use non-standard options and features do you need to purchase the source code. Changes in the compiler flags, which have been carefully optimized for performance and robustness, are not recommended and the use of different compiler flags is at your own risk.

Scientists interested in further developing the Amsterdam Modeling Suite should e-mail (info@scm.com) us to discuss a possible collaboration (https://www.scm.com/projects-collaborations/) with SCM, including a discounted (source code) license.

17.12 What level of support can I expect?

Customers recommend us (https://www.scm.com/adf-modeling-suite/adf-users-say) for the level of support and expertise of our staff. At SCM, the people developing the code are the people giving technical support.

This includes all technical aspects related to installation and bugs, and we sometimes also give some more scientifically oriented help. The latter is done only at SCM's discretion and is not included in the official standard support.

While we sometimes like to help inexperienced users to get started, we do expect researchers to get a grasp of the theoretical methods and relevant literature. You could also reach out to the scientific community through our ADF mailing list and other channels.

We cannot guarantee to fix a particular problem or answer any question, but we will do our best to help you!

17.13 Should I use SMT/hyperthreading or only physical cores? What about E and P cores?

Hyperthreading does not speed up your ADF (or BAND, DFTB, ReaxFF) calculations, so it is not recommended to enforce this (by using a license for more cores and setting **NSCM** to double the number of physical cores). For licensing purposes, we only count physical cores (https://www.scm.com/adf-modeling-suite/pricing-licensing/counting-processors/). So, for instance, a hexa core machine with hyperthreading counts as six cores.

Similarly, starting from AMS2023 we do not count the E-cores for processors that use the big.LITTLE architecture where fast P cores are combined with slow but energy-efficient E cores. For such processors only the physical P cores are counted.

The Bulldozer-family of AMD architectures (Bulldozer, Piledriver, Steamroller) have only one floating point unit per two cores (module). It is thus recommended to run ADF only on half of the cores on these machines, and we try to license these machines accordingly.

For the AMD Zen generation CPUs (Ryzen/Threadripper/Epyc) ADF should run on all physical cores (without SMT). Starting with AMS2019 special Zen-optimized binaries are available that have been optimized for this CPU architecture.

17.14 How can I limit the number of processors or CPU cores used by AMS or one of the modules?

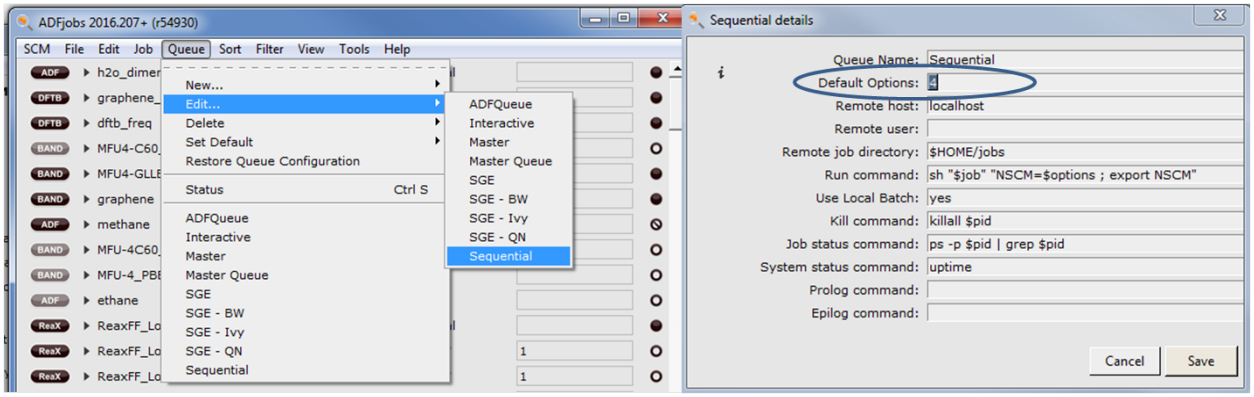
By default the Amsterdam Modeling Suite and its engines try to use all physical cores. So no hyperthreading and half of the cores of the some AMD architectures. To change this behavior, you need to set the **NSCM** variable, either in the environment, in your submit script, or in the queue definition in AMSjobs.

E.g. running from the command line on Linux:

```
export NSCM=4
```

will make ADF and other modules use 4 cores even if there are 2 or 8 physical cores.

Likewise in AMSJobs you can change your sequential queue **Queue => Edit => Sequential**. Enter the desired number of cores in the **'Default Options:'** field. Since this is the default queue so if you run a job without specifying a queue or modifying the default field in AMSJobs, it will use your indicated number of cores.



17.15 What does MPI Application rank ... exited before MPI_Finalize() with status ... mean?

MPI Application rank … exited before MPI_Finalize() with status … is an error that could have occurred for several reasons. Please contact support@scm.com with:

- input (.run), output (.out), logfile (.log, error (.err) for your job
- your operating system (Windows 32/64 bit, Linux (distribution), MacOS (version))
- which version of AMS you downloaded

17.16 How do I run and monitor remote jobs remotely?

It is easy to have your local GUI to set up and monitor jobs as well as visualize results, while submitting jobs to remote queues.

A video shows how to set up remote queues on Windows (https://www.scm.com//wp-content/uploads/Videos/RemoteQueuesWithADFJobs.mp4).

There are some conditions that must be fulfilled however. The main condition is that you must be able to log on to the remote computer using ssh (or, on Windows, plink.exe) **without being asked for any passwords or confirmations**. The common way for this is using public key authentication and the ssh-agent. Detailed instructions for Windows users are available in the Readme.rtf document and in its PDF version DocInstallation_windows.pdf (both in the AMS installation folder). Unix/Linux users probably already know how to do this. The other condition is: AMS must be installed on the remote machine and the commands that set up relevant environment variables must be present in the shell profile file, for example ~/.profile.

Suppose you are able to log on to the remote machine and suppose the machine is called cluster.example.com, and your login there is myname (of course you will use real values instead of these). The machine is a login node of a cluster and PBS is the batch system used on the cluster. Our experience shows that this is a very common configuration among AMS users. Now we need to configure a new queue in AMSjobs. Let's call the queue cluster_PBS. Use the menu command **Queue->New…->PBS**. The template already supplies some default values but you may want to change them depending on the cluster policies.

First the easy parts: type cluster_PBS in the Queue Name field, cluster.example.com for Remote host, myname for Remote user. The Remote job directory specifies a directory on the cluster where AMSjobs will place job files.

The Run command is probably what you will want to think about more thoroughly. In the template it contains

```
qsub -lnodes=2:ppn=2:infiniband -lwalltime=$options "$job"
```

You may have to change everything between qsub and "$job" on the line depending on how PBS is configured on the cluster. This will also determine the value for the Default Options entry.

In the template, the wall clock time is configurable per job but you may decide that it is better to have the number of nodes as an option. In this case the Run command will look as:

```
qsub -lnodes=$options:ppn=2:infiniband -lwalltime=100:00:00 "$job"
```

Of course you can add there other qsub command line switches that you would normally put in the script after #QSUB. So if you choose to "hard-code" the number of nodes and leave walltime configurable then you may want to name the queue something like cluster_PBS_2nodes. If you choose to hard-code the walltime and leave the number of nodes configurable then you may want to name the queue cluster_PBS_100hr or something like that so that you can distinguish it from other similar queues that differ from this one by the hard-coded value.

After you've finished configuring the queue, press Save and test that you can see the status of the queue by selecting menu Queue->Status. This will query the status for each queue defined in AMSjobs by running the command specified in the System status command field (remotely if necessary). If you do not get the queue status but something like "Connection refused" or "No more authentication methods left" then check the ssh connection to the remote computer.

Finally test your queue by running a Test job.

If everything works as you like, you may consider setting your new queue as the Default Queue in AMSjobs. Then using the Run command in AMSinput or in AMSjobs will automatically assign this new queue to your job, and run the job on the remote machine.

17.17 I get a 'libGL error' mentioning 'libstdc++' in AMS2021 or older

This is error is caused by the GUI trying to use the packaged libstdc++, which is older and incompatible with your system libstdc++ needed for your graphics drivers.

The solution is to move the packaged libstdc++. This is done with:

```
mkdir $AMSBIN/libold
mv $AMSBIN/lib/libstdc++* $AMSBIN/libold/
```

From AMS2022 onwards this issue is solved with a dependency checker.