Installing Software on High Performance Computing Systems

John Zaitseff
Research Technology Services
June 2021
1. Research Computing
   - High Performance Computing: Gadi, Katana, …
   - Cloud computing: Amazon AWS, Microsoft Azure, Google
   - Code and algorithm support

2. Research Data
   - Data management including highly sensitive or complex data
   - Assistance with data moves, storage, planning, tools
   - Pilot scheme for publishing Open Data
   - UNSW GitHub private, public and limited-sharing repositories
3. Research Community

- Over 50 free training courses: Linux, Python, Matlab, R, …
- Weekly Hacky Hour meetings: via Microsoft Teams, on Thursdays at 3pm. Bring your problems with code, HPC, data and more!
- ResTech seminars, lunch-and-learn series, training videos, …

4. ResBaz

- Annual data and compute literacy festival/conference for researchers from all over New South Wales
- Watch this space for 2021!
Installing Software on HPC Systems

- Assumed knowledge
- Avoiding software installations
- Installing binary packages
- Installing from source code
- Creating module files
- Installing Python packages
- Installing R and RStudio libraries
- Installing Julia packages
- Questions?
Assumed knowledge

- You have an account on a High Performance Computing system
  - Katana at UNSW
  - Gadi at NCI
- You know how to log in to that HPC system via SSH (Secure Shell)
  - See the Katana documentation or NCI Help pages for details
- You know basic Linux commands
  - See the Introduction to Linux and High Performance Computing course notes and associated recorded video
- You understand basic module commands
  - See Software Modules part 1 and part 2
- You are not afraid to try doing things yourself!
Avoiding software installations

First rule of installing software on HPC: **DON’T**

- Is your software already installed system-wide?
- Is your software already installed by your colleagues?
- Is a different version of your software already installed?
- Is a similar package already installed?

- Try asking HPC staff to install your software for you
  - For Katana: send an email to [itservicecentre@unsw.edu.au](mailto:itservicecentre@unsw.edu.au) mentioning Katana
  - For Gadi: send an email to [help@nci.org.au](mailto:help@nci.org.au)
Checking if a package is installed

- Is your required package already part of the base operating system?
  - Default editors, programming languages, compilers, debuggers and libraries
  - Check using `yum list installed` (ignore any warnings about old repositories)

- Is it already installed system-wide?
  - Multiple versions of applications, all stored in `/apps` directory
  - Check using `module avail`

- Is it installed by your colleagues?
  - Ask them!

Try it now:

```
yum list installed | less  # To return to the command line, press “q” to quit
module avail |& less      # … press “q” to quit. Note the unusual “|&”!
```
Installing binary packages

• Compiling from source code is often best
  – Allows you to use specific HPC-optimised compilers, compiler flags and libraries

• If only a binary package is available:
  – Download the binary package (avoid DEB and RPM packages): 64-bit Intel x86_64 / AMD64 architecture, CentOS 7 compatibility for Katana, CentOS 8 for Gadi
  – Start an interactive session using `qsub -I` with appropriate parameters
  – Follow the supplied instructions to install the software
  – Do not try to use `apt-get install`, `yum install`, `dnf install`, `su` or `sudo`!

Try it now:

```
chmod a+rx ./trader-7.16-x86_64.AppImage # Make the application executable
./trader-7.16-x86_64.AppImage --help # Test the application
```
Installing from source code

- Download the source code
- Download source code to any library dependencies
- If possible, use the system compiler and linker (gcc and ld) and libraries
- Start an interactive session using `qsub -I` with appropriate parameters
- If necessary, load any required modules for compilers and libraries
- Follow the supplied instructions for compiling the package
- Install the software to your home directory or scratch directory (for large packages)
- With Autoconf-enabled software, often as simple as running
  
  ```
  ./configure --prefix=$HOME/apps/PACKAGE/VERSION
  make && make install
  ```
Installing from source code

Try it now on Katana:

```bash
mkdir ~/src; cd ~/src       # Source code will be stored in $HOME/src
  # Download the source code

tar xvf trader-7.16.tar.xz  # Unpack the source code
cd trader-7.16             # Change to the source code directory
less README; less INSTALL  # Read the installation instructions; "q" to quit each file
qsub -l walltime=0:30:00 -l select=1:ncpus=1:mem=8GB -l        # Request an interactive job (you may need to wait)

Once a command line prompt appears:

cd ~/src/trader-7.16       # Go to the source code directory
./configure --prefix=$HOME/apps/trader/7.16    # Configure the software
make && make install      # Compile and install the code
~/apps/trader/7.16/bin/trader --help       # Test the installed software
cd ~/src; rm -fr trader-7.16       # Remove the source code to save space
```
Installing from source code with modules

Try it now on Gadi:

```
mkdir ~/src; cd ~/src # Source code will be stored in $HOME/src
tar xvf trader-7.16.tar.xz # Unpack the source code
qsub -l walltime=0:30:00 -l ncpus=1 -l mem=8GB -I # Request an interactive job (you may need to wait)
```

Once a command line prompt appears:

```
cd ~/src/trader-7.16 # Change to the source code directory
module load intel-compiler/2021.2.0 # Use the Intel compiler (icc)
./configure --prefix=$HOME/apps/trader/7.16 # Configure the software
make && make install # Compile and install the code
~/apps/trader/7.16/bin/trader --help # Test the installed software
```

```
cd ~/src; rm -fr trader-7.16 # Remove the source code to save space
```
Creating module files

• The environment module system allows for multiple versions of applications

• Short “recipes” for how to modify your compute environment
  – On Katana, stored in /apps/modules
  – On Gadi, stored in /apps/Modules (note capital “M”!)
  – Sample file /apps/modules/templates/module_file on Katana

• Written in the TCL programming language

• Many powerful features!
  – See documentation by running man module and man 4 modulefile

• Can easily create and use your own module file recipes
  – Create a directory ~/apps/Modules
  – Add “module use --append $HOME/apps/Modules” to your ~/.bashrc file
  – Log out and log back in to enable the module use command
Creating module files

Try it now on Katana or Gadi:

```
mkdir ~/apps/Modules  # Module files will be stored here
mkdir ~/apps/Modules/trader  # Module files for the Star Trader application
nano ~/apps/Modules/trader/7.16  # Create the module file
```

Enter the following text inside the Nano editor:

```
#%Module1.0
set             basepath        $env(HOME)/apps/trader
set             version         7.16
set             path            $basepath/$version
prepend-path    PATH            $path/bin
prepend-path    MANPATH         $path/share/man
```

Press **CTRL-X** to save the file and exit the editor (follow the prompts on the bottom of the screen)
Using custom module files

Try it now on Katana or Gadi:

```bash
module use --append ~/apps/Modules
# Custom module files are stored here
# … can be added to your ~/.bashrc file

module avail trader
# Is the Star Trader application available?
module load trader/7.16
# Use the new module file

trader --help
# Test the application: no need for a full path
man trader
# Manual page is also available ("q" to quit)
```
Installing Python packages

• On Katana, many Python packages are already installed
  – Check by running `python3 -c "import PACKAGENAME"` after running `module load python/VERSION` (for an appropriate 3.x.y version)
  – Can also run `pip3 list` to list package version numbers
• If not available, create a Python Virtual Environment and install the required Python package
• If required, install Conda or Anaconda for yourself—these are not able to be installed for multiple users
• Full instructions are available in the Katana Python documentation
• Python Virtual Environments can be used from Katana On Demand
  – See the Katana Jupyter Notebooks documentation for details
Installing R and RStudio libraries

• On Katana, many R and RStudio libraries are already installed
  – Load the appropriate module file: `module load R/4.0.2`
  – Note that `module load R/4.0.2-clean` loads a version of R that does *not* contain additional libraries! Useful if you want to install newer versions of libraries that conflict with those in R/4.0.2.
  – Start R, then run `library()` to check available libraries
• To install an R library, download the package and run `install('PACKAGE_PATH')`
• Further instructions are available in the Katana R and RStudio documentation
Installing Julia packages

- Only the default Julia packages are installed
- To install or update a package, use the Julia Package Manager (`Pkg`)
  - Load the appropriate module file: `module load julia/1.6.1`
  - Start Julia from the command line: `julia`
  - Enter the package manager: Press "\]
  - Refresh the package list: `up`
  - Add your required packages: `add PACKAGE`
  - Exit the package manager: Press `CTRL-C`
  - Exit Julia: `exit()`

- You can now use the new packages in your Julia code
5. With whom do I discuss my HPC needs?

1. Your colleagues
2. Your supervisor
3. Hacky Hour: every Thursday 3pm on Microsoft Teams (Research Technology Training, Hacky Hour channel)
4. The Research Technology Services team
   - John Zaitseff
     J.Zaitseff@unsw.edu.au
   - The whole team at UNSW
     restech@unsw.edu.au

https://restech.unsw.edu.au/