Pawsey

Using Modules and Containers

Setonix Phase 1 Release

20 June 2022. Version 1.01

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

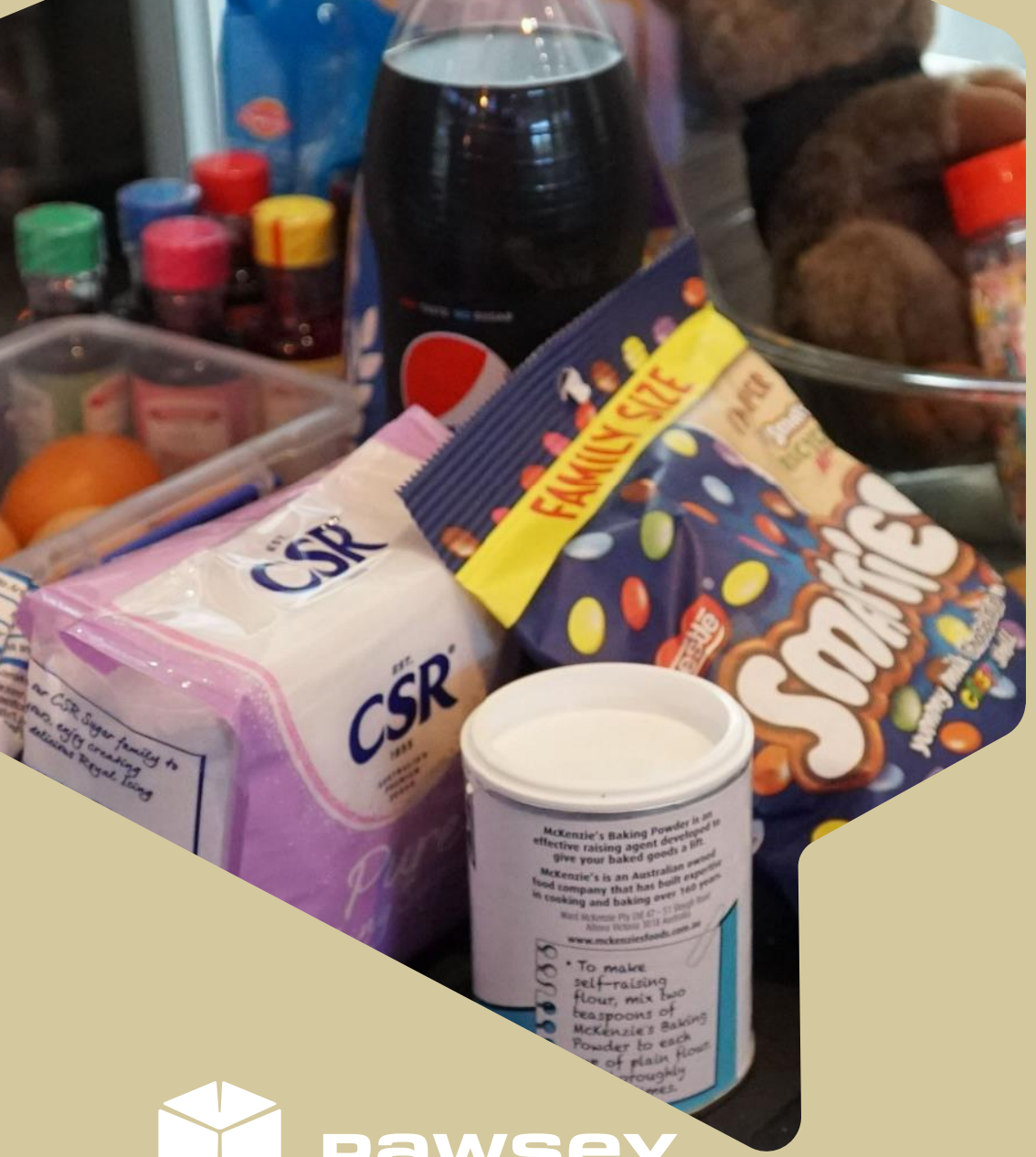
Focus for This Training

Learning Outcomes:

- Summarise changes to existing practices to using modules and containers on Setonix.
- Determine whether and how API libraries and driver changes impact your project.
- Discuss updates to loading modules when using Setonix.
- Identify when software installations are necessary to migrate to Setonix (see module 04 for how to install software).
- Understand the programming environments that a module is compatible with.
- Summarise changes to your container use when using Setonix.

Core Migration Training Modules:

1. Getting Started with Setonix
2. Supercomputing Filesystems
3. Using Modules and Containers
4. Installing and Maintaining Your Software
5. Submitting and Monitoring Your Job
6. Using Data Throughout the Project Lifecycle



pawsey

Section 1

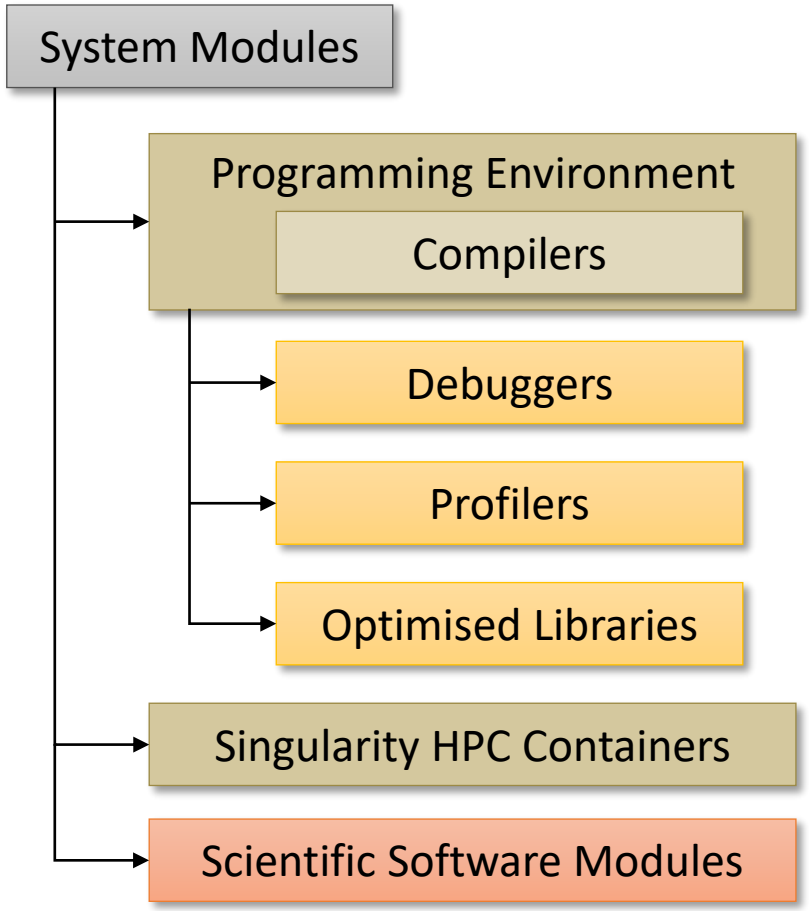
Types of Software on Setonix

Software Stack Overview

Software stack includes vendor-supplied operating systems, compiler suites, high-performance libraries and research-domain-specific packages.

Sources of Software

- HPE/Cray provides optimised compilers, libraries and tools.
- Pawsey supports a core of optimised scientific applications and libraries commonly adopted by many user groups.
- Project groups are responsible for installing and maintaining domain-specific software required for their computational workflows.



Location of Software

- Software stack on Setonix is located on `/software` in 3 levels:

  System-wide: Accessible by all users; access is by the Module system; maintained by the Pawsey staff. Use this software if possible as these are optimised for Setonix hardware. Resides under: `/software/setonix`

  Project-wide: Maintained by project members; accessible by all members; resides in: `/software/projects/`*`project-id`*

  User: Maintained by user (for example, to download containers). Recommended path: `/software/projects/`*`project-id/user-name`*

- Details available @
  - Software Stack
  - Software Stack Policies

Software: Key Changes

- Moving from Intel architecture to AMD means a move from Intel compilers to AMD and Clang-based compilers.
- Pawsey-provided software modules will be initially built primarily with PrgEnv-gnu (now the default environment).
- Move from Maali to Spack installation tool.
- Some system-wide installations (bioinformatics and OpenFOAM) are performed as Container Modules, by means of SHPC.
- Move from Environment Modules to Lmod modules.
- Versions should be specified for all modules.

Available Software on Setonix

Not an exhaustive list

Languages and Parallel Models	Programming Environments	Optimised Libraries	Development Tools	Supported Software
C/C++ Fortran	Cray: Clang-based C/C++, Cray Fortran	Mathematical Libs: BLAS, LAPACK, ScaLAPACK, FFTW	Debuggers: gdb4hpc, ARM DDT, Valgrind4hpc	Libraries: PETSc, Boost, Trilinos Plumed, OpenCV, Slate, HPX, Kokkos
Python, R, Perl, Ruby, Java, Rust	GNU: GNU C/C++, Fortran	I/O Libs: NetCDF,ADIOS,HDF5	Profilers: Cray Perftools, ARM Map	Tools: Singularity, NextFlow VisIT, ParaView, VMD, HPC Python
Distributed Memory: MPI	AOCC: AMD-provided Clang-based C/C++		Code Development: Reveal, CCDB	Domain: Gromacs, NAMD, LAMMPS, NWChem, Fluent, OpenFOAM, VASP, ANSYS, Nektar
Shared Memory/GPU: OpenMP, Pthreads			Build & Install: Make, CMake, Spack, Pip	

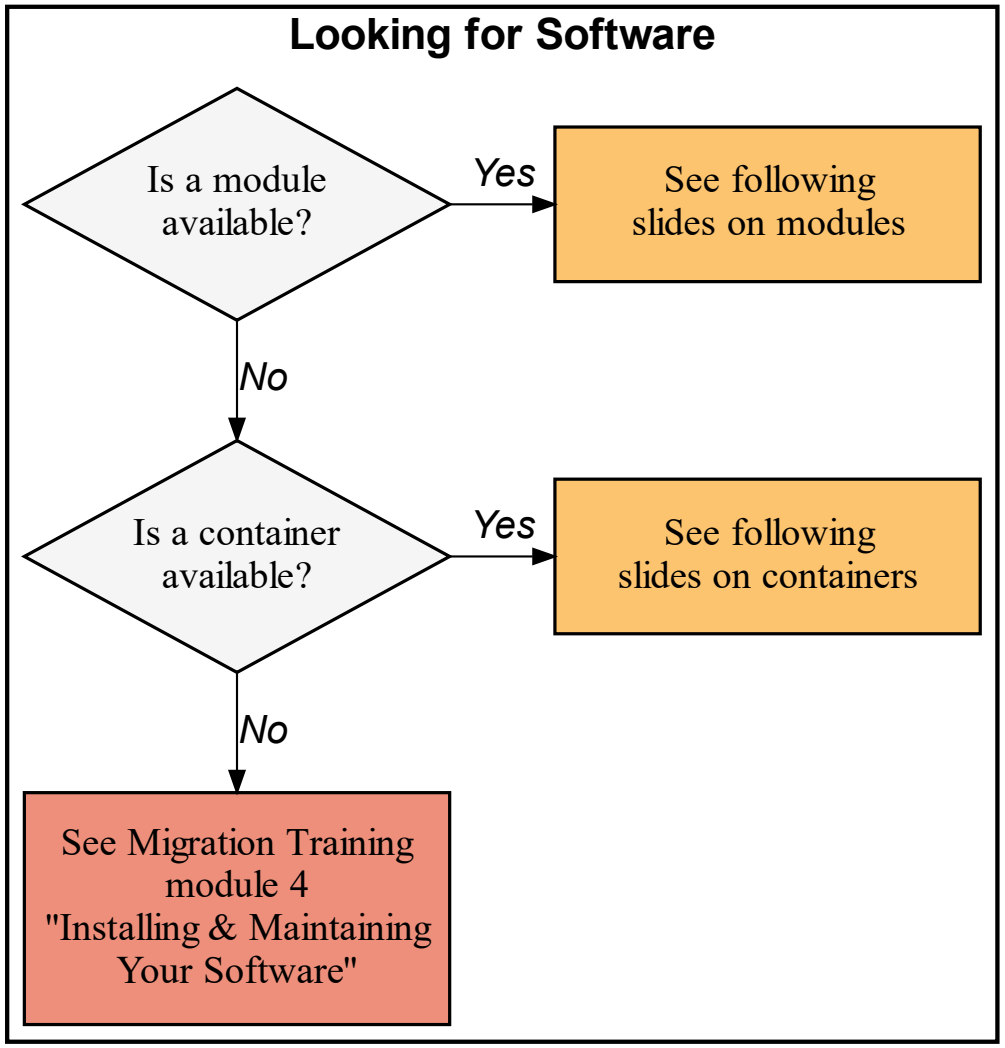
Details available @

- Popular Domain-Specific Software
- List of Supported Software

Is the software you need available on Setonix?

1. Is a module available?
   - **Yes** – see Modules on Setonix (next section in this training)
   - **No** – go to step 2
2. Is a container available?
   - **Yes** – see Containers on Setonix (later in this training)
   - **No** – see Migration Training Module 04: Installing and Maintaining Your Software*
     - Installing using Package Managers (Spack, Pip, Conda, R, SHPC)
     - Full manual installs

*More information on Migration Training Module 4: Materials, Recordings



pawsey

Section 2

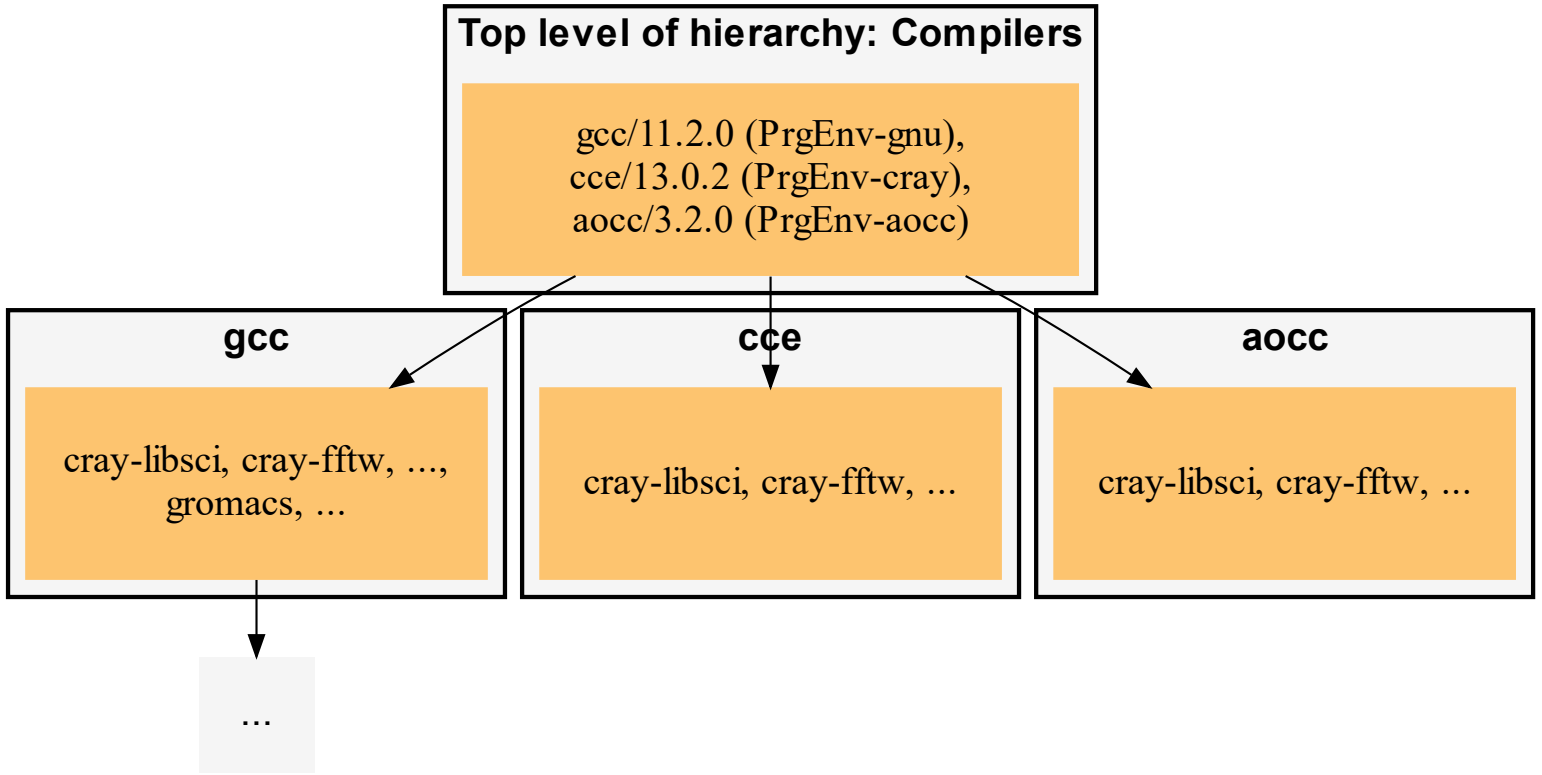
Modules on Setonix

Setonix Module System: Lmod

- Setonix uses Lmod.
- Magnus and Galaxy used Environment Modules;
  Topaz, Zeus, Garrawarla and ASKAP Ingest also use Lmod.
- Lmod hierarchies may hide modules from you, depending on loaded compilers.
- Lmod reloads dependent modules (such as compilers and libraries) for you when swapping a module.
- Modules will not have default versions:
  - Use `module load` *`package/version`*
- Some commands are different:
  - `module avail` – can use wildcards (*), can search for substrings
  - `module spider` – can list modules that can be loaded in the system across compiler hierarchies
- Command output format may differ for show, whatis, help.

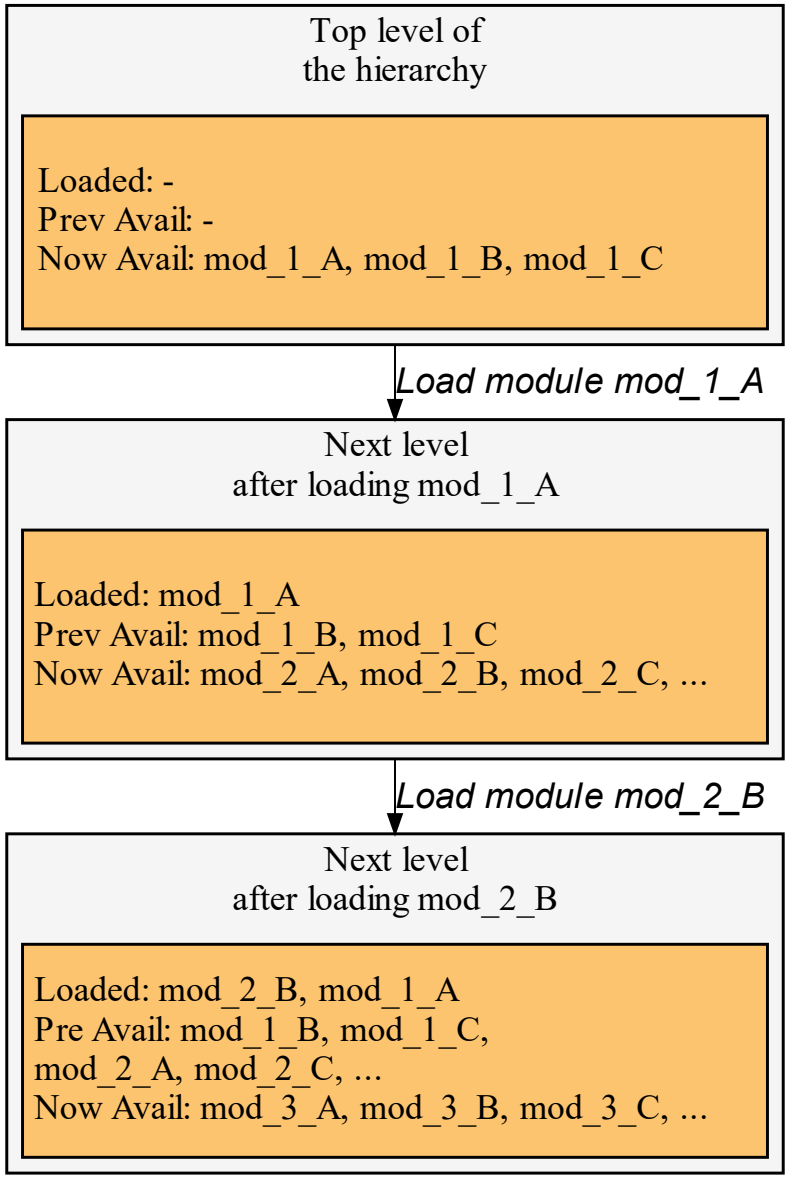
Lmod Hierarchy

- Modules are organised in a hierarchy.
- Top level consists of compilers accessed by Programming Environment modules `PrgEnv-`*`compiler`*
- Other modules at subsequent levels of the hierarchy are only available once the previous level is loaded.
- Set of modules along each branch may differ.



Using Lmod Hierarchy

1. Choose one of the available modules at top level of hierarchy. On Setonix this will be a Programming Environment.
   - For example, `PrgEnv-gnu`
2. This will make available for loading new modules that exist on the system.
   - i.e. in the `module avail` list
3. You can then load these new modules. New modules may provide access to deeper levels of the hierarchy.



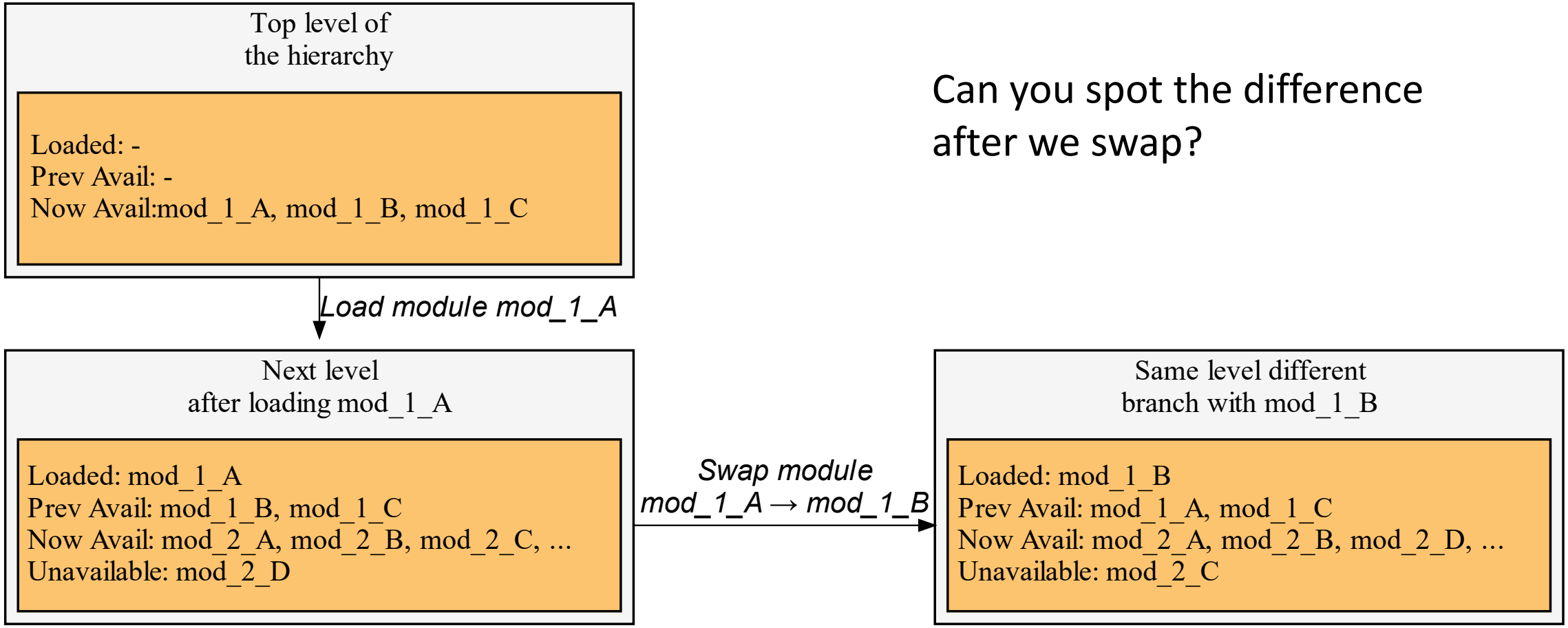
Hierarchies in Action

`avail` does NOT list modules that cannot be loaded

`spider` shows all modules that match *string* and what must be loaded to make each module visible

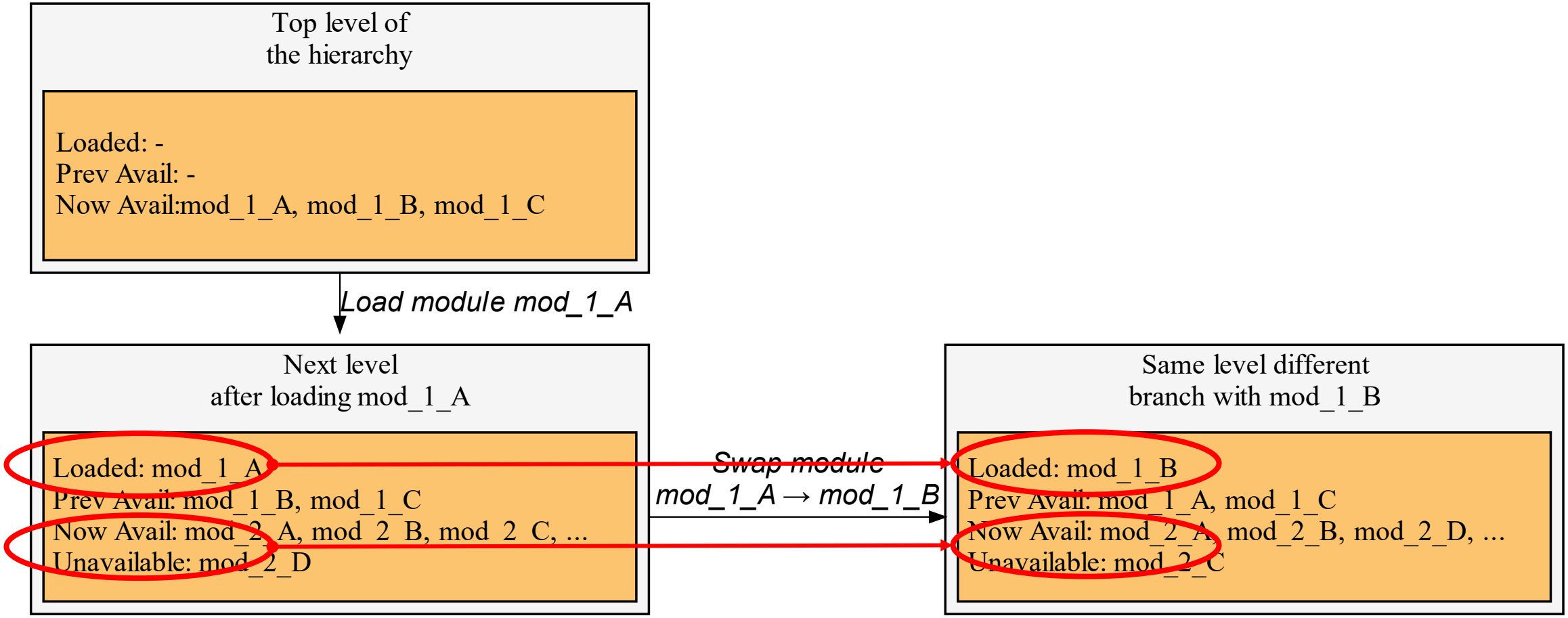
```
$ module avail hdf5 #module command avail does not list all modules that can in principle be loaded
--------------------------- /opt/cray/pe/lmod/modulefiles/mpi/gnu/8.0/ofi/1.0/cray-mpich/8.0 -----------------------
   cray-hdf5-parallel/1.12.1.1
------------------------------------------------- /opt/cray/pe/lmod/modulefiles/compiler/gnu/8.0 -------------------
   cray-hdf5/1.12.1.1
----------------------------------- /software/setonix/current/modules/zen3/gcc/11.2.0/libraries/ ---------------------------
 adios2/2.7.1-hdf5            hdf5/1.12.1-api-v110          hdf5/1.10.7-api-v18          hdf5/1.12.1-api-v112
 …
-----------------------------------  /software/setonix/current/containers/modules ----------------------------------------
 hpc-python-container/2022.03-hdf5mpi

$ module –r spider hdf5* #module command spider walks all branches of the hierarchy searching for modules that contain the
substring
----------------------------------------------------------------------------------------------------------------------------
  cray-hdf5: cray-hdf5/1.12.1.1
----------------------------------------------------------------------------------------------------------------------------
    You will need to load all module(s) on any one of the lines below before the "cray-hdf5/1.12.1.1" module is available to load.
      aocc/3.2.0
      gcc/10.3.0
      …
----------------------------------------------------------------------------------------------------------------------------
  cray-hdf5-parallel: cray-hdf5-parallel/1.12.1.1
----------------------------------------------------------------------------------------------------------------------------
    You will need to load all module(s) on any one of the lines below before …
      aocc/3.2.0  cray-mpich/8.1.14
      cce/13.0.2  cray-mpich/8.1.14
      craype-network-none  cray-mpich/8.1.14
      craype-network-ofi  cray-mpich/8.1.14
      …
```

Swapping Branches in the Hierarchy: Spot the Difference



Can you spot the difference after we swap?

Swapping Branches in the Hierarchy: Revealed



Example of Swapping

Load module

Swap Env

Loading a module that exists but is not available

```
$ module list
Currently Loaded Modules:
  1) craype-x86-milan         5) xpmem/2.2.40-2.1_2.56__g3cf3325.shasta  9) cray-libsci/21.08.1.2
  2) libfabric/1.11.0.4.75    6) gcc/11.2.0                              10) cray-mpich/8.1.14
  3) craype-network-ofi       7) craype/2.7.14                           11) PrgEnv-gnu/8.3.2
  4) perftools-base/21.12.0   8) cray-dsmml/0.2.2                        12) pawsey
$ module load cray-hdf5
$ module list
Currently Loaded Modules:
  1) craype-x86-milan                      6) gcc/11.2.0           11) PrgEnv-gnu/8.3.2
  2) libfabric/1.11.0.4.75                 7) craype/2.7.14        12) Pawsey
  3) craype-network-ofi                    8) cray-dsmml/0.2.2     13) cray-hdf5/1.12.1.1
  4) perftools-base/21.12.0                9) cray-libsci/21.08.1.2
  5) xpmem/2.2.40-2.1_2.56__g3cf3325.shasta 10) cray-mpich/8.1.14

$ module swap PrgEnv-gnu/8.3.2 PrgEnv-cray/8.3.2
Due to MODULEPATH changes, the following have been reloaded:
  1) cray-hdf5/1.12.1.1     2) cray-mpich/8.1.14

$ module load cray-netcdf-hdf5parallel/4.8.1.1
Lmod has detected the following error:  These module(s) or extension(s) exist but cannot be
loaded as requested: "cray-netcdf-hdf5parallel/4.8.1.1"
```

Setonix Lmod Hierarchy

- Pawsey-supported modules have implemented a hierarchy at just the Programming Environment (compiler) level
    - Non-Pawsey supported Cray modules use a deeper, more complex hierarchy. Example: Cray provided IO libraries such as HDF5 use a deeper hierarchy.

```
$ module avail python/3.10.4
------ /software/setonix/current/modules/zen3/gcc/11.2.0/programming-languages ---------
  python/3.10.4
$ module spider python/3.10.4
----------------------------------------------------------------------------
  python: python/3.10.4
----------------------------------------------------------------------------
  You will need to load all module(s) on any one of the lines below before the "python/3.10.4" module is
available to load.
      aocc/3.2.0
      cce/13.0.2
      gcc/11.2.0
```

Runtime Reproducibility with Modules

- Since programs dynamically load dependencies at runtime, the availability of dependencies through modules may impact runtime behaviour.
- Search for dependencies differs between Pre-Setonix systems and Setonix:
  - Pre-Setonix:
    - Executable at runtime searches for libA , first instance in library paths found will be used, which is defined by loaded modules.
  - Setonix:
    - Executable at runtime searches for specific path, `/path/to/a/libA`, which is defined at build time.
- The outcome is a more reproducible runtime.

Modules: Pre-Setonix

- Magnus used Environment Modules, which does not have a hierarchy of modules that work together. Rather module files have explicit list of conflicts with other modules.
- Modules that provide the same functionality through a library can be loaded at the same time if there is no explicit conflict listed. *This means that module order can impact behaviour due to the use of dynamic libraries.* Consider this example:
  - **A** depends on functionality that can be provided by either library **B** or **C**.
  - **A** loads by default **B**.

  Four possible scenarios are:

Scenario 1	Scenario 2	Scenario 3	Scenario 4

`module load A`	`module load B`
	`module load A`
		`module load A`
		`module load C`
			`module load A`
			`module load C`
			`module load B`

`Result: A+B`	`Result: A+B`	`Result: A+C`	`Result: A+B`

Modules: Pre-Setonix (cont.)

- Magnus used Environment Modules, which does not have a hierarchy of modules that work together. Rather module files have explicit list of conflicts with other modules.
- Modules that provide the same functionality through a library can be loaded at the same time if there is no explicit conflict listed. *This means that module order can impact behaviour due to the use of dynamic libraries.* Consider this example:
  - **A** depends on functionality that can be provided by either library **B** or **C**.
  - **A** loads by default **B**.

  Four possible scenarios are:

C overwrites B

C overwrites B
then B overwrites C

Scenario 1	Scenario 2	Scenario 3	Scenario 4

`module load A`	`module load B`
	`module load A`
		`module load A`
		`module load C`
			`module load A`
			`module load C`
			`module load B`

`Result: A+B`	`Result: A+B`	`Result: A+C`	`Result: A+B`

Modules: Setonix

- The use of Lmod with a module hierarchy combined with the use of Spack to build software means that the scenario in the previous slide cannot happen.
  - Conflicts are handled with modules residing in different hierarchies.
  - Executables directly refer to the library that they are compiled with.
  - Modules on Setonix have pre-determined dependencies that cannot be changed at runtime.
- If software **A** was built with library **B**, it will always have that behaviour regardless of other modules loaded, even if library **C** provides similar functionality. Example:

```
module load anything
module load A # implies loading optimal software A + dependency B
module load anything
```

Result: A+B

Key Change in Use of Modules

- Modules on Setonix have predetermined dependencies that cannot be changed at runtime.
- This reduces flexibility somewhat but it makes it:
  - More reproducible
  - Less error prone
  - More transparent
- You can view predetermined dependencies with:
  `module show` *`module-name/version`*
  (the module must be available to be loaded)

Programming Modules on Setonix

Summary

- A programming environment should be loaded as a dependency for other modules.
  - The PrgEnv-gnu environment is loaded by default
- The availability of modules will now depend on the loaded programming environment module.
  - `module spider` will show all available modules for all programming environments

More information

- See Migration Training Module 04: Installing and Maintaining Your Software (Materials, Recordings) for more details on:
  - Changes to compiler usage
  - Complete list of programming languages with compilers or interpreters available

⚠ **NOTE:** Python 2 is not supported on Setonix.

Supported Scientific Software: Key Changes: Additions

Adding Support (not exhaustive list)

Category	Packages
HPC Applications	CP2K, Nektar, WRF, ROMS
API Libraries	Adios, HPX, Kokkos, OpenCV, Plasma, Scalapack, Slate
Programming languages	Rust
Python libraries	Dask, H5netcdf, Numba, Plotly, Scikit-learn
Astronomy	WCSTools, WSClean
Bioinformatics	BBtools, Diamond, ExaML, GATK, Maker, Sambamba

- Check the full list of supported software @ Popular Domain-Specific Software

Supported Scientific Software: Key Changes: No Longer Supported

Modules no longer provided

Category	Package	Notes
HPC Applications	Moose	Low uptake
	SIESTA	Low uptake
Programming languages	Python 2	End of Life

- Check the full list of supported software @ Popular Domain-Specific Software

Further References for Modules

- Check supported software @ Popular Domain-Specific Software
- Software updates provided via Pawsey Technical Newsletter
- More details available @ Modules

Section 3

Containers on Setonix

pawsey

When to Use Containers

- Containers package an application and all of its dependencies into a single, sandboxed piece of software:
  - Pre-installed software and dependencies reduce installation time.
  - Sandboxed software greatly improves reproducibility and portability across environments and infrastructures.

- Containers are advantageous in certain circumstances:
  - Large volume of software to be installed (e.g., bioinformatics)
  - Complex builds with lots of dependencies (e.g., deep learning frameworks)
  - Dependency trees with high changes of package/version conflicts (e.g., Python and R workflows)
  - Need to mitigate intensive I/O patterns in software that writes a large number of small files (e.g., OpenFoam)

Using Containers on Setonix

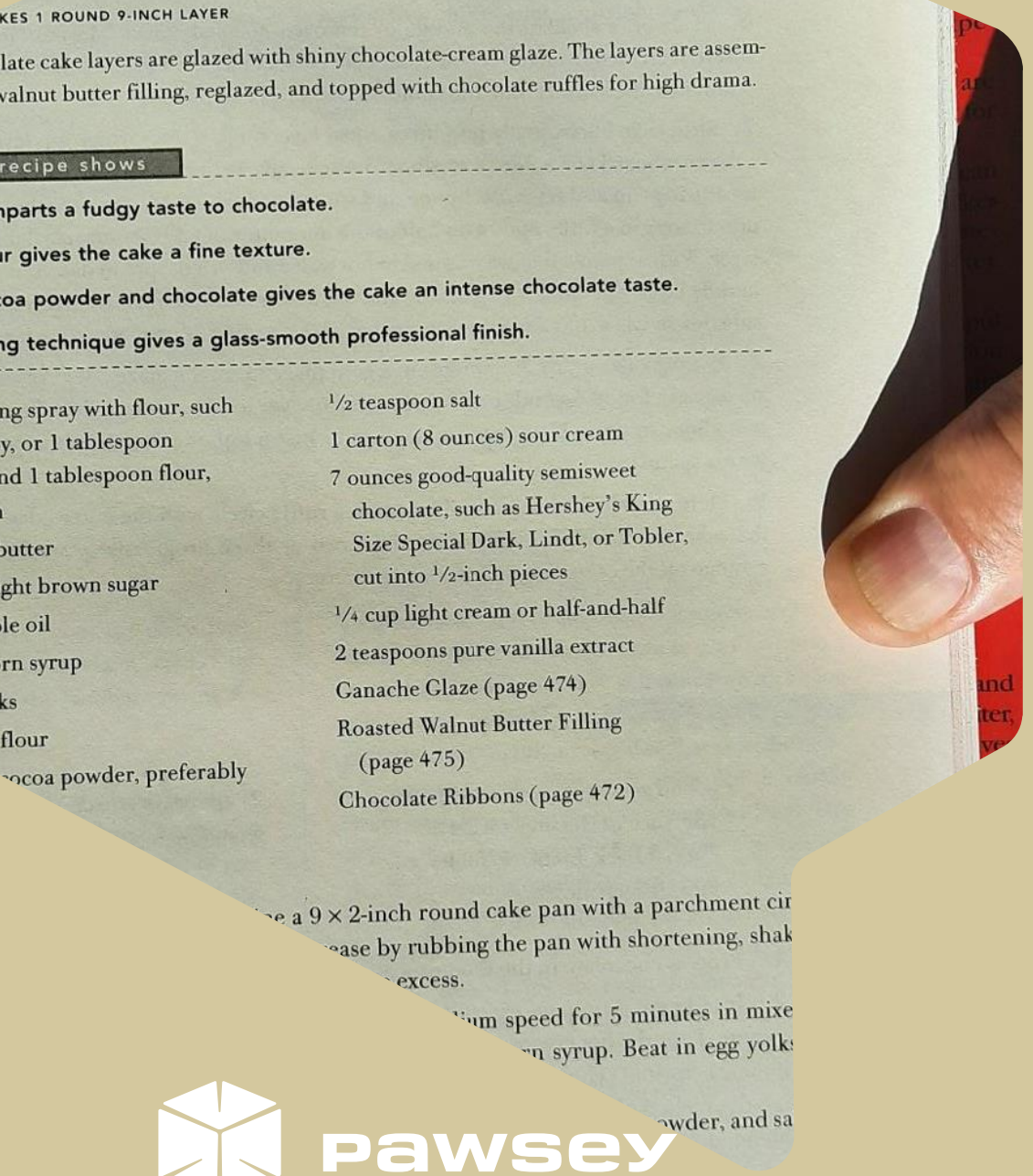
- Singularity is the container runtime on Setonix.
- Already available as system-wide module and configured for Setonix – do not install your own version!
  - There is also a singularity-astro container for astronomy users
- The module sets environment variables for default behaviours
  - To expose `/scratch` and other relevant filesystems in the containers
  - To ensure MPI applications in containers make use of the high-speed interconnect
- Containers on Setonix can be used in two ways
  - Same as pre-Setonix, via the Singularity commands
  - NEW: transparent, module-like experience, when containers are installed as Container Modules via SHPC

```
# Singularity commands
$ module load singularity/3.8.6
$ singularity exec container.sif
cmd args

# Container Modules
$ module load tool/version
$ tool cmd args
```

Container Modules with SHPC

- Singularity Registry HPC (SHPC) is a tool to automate installation of containers as *container modules*.
- Container modules allow the containerised application to run using the same commands as its non-containerised counterpart.
- This interface is implemented by either shell scripts (on Setonix) or shell functions that wrap and hide the Singularity container runtime syntax.
- Most bioinformatics packages on Setonix are installed as container modules.

- More details @ https://singularity-hpc.readthedocs.io/en/latest/



pawsey

Section 4

Software Summary

Considerations and Policies

- Pawsey provides and supports popular software from multiple scientific domains. Use this software via modules, if possible, as these are optimised for Setonix hardware.
  - Pawsey does not pay for licensed domain software, which is left to users.
  - Pawsey does provide some popular HPC applications that have a restricted license (Bring Your Own License).
- Pawsey provides documentation for users to build and install software that is not supported by staff.
  - Additional assistance in installing unsupported software is discretionary and evaluated case by case, with priority given to PaCER, strategic and uptake projects.
- Refer to the Software Stack Policies page for more details.

Summary of Key Changes with Impacts

Topic	Existing	New	Impact
Modules	Environment Module	Lmod	More reproducible, better conflict resolution
Package manager	Maali	Spack	Move to community developed tool, well documented, human-readable CLI
Compilers	Cray, GNU, Intel	Cray, GNU, AOCC	Intel compilers replaced by AMD optimised compilers. Cray compilers now based on Clang. Software compiled with Intel no longer available.
Scientific software		Updated software	Pawsey supported list has minor changes with most previously supported software still supported.
Containers	Singularity	Singularity, SHPC	Option for module-like user experience by means of Container Modules.

How do I get help?

Migration Documentation & Migration Guides

- Setonix Migration Guide
- Setonix User Guide
- Supercomputing Documentation

Migration Training Materials & Video Recordings

- Upcoming Migration Training
- Recordings: Pawsey YouTube Setonix Migration Phase 1 Playlist
- Materials: Setonix Migration Training Materials (PDFs)

Help Desk

- Help Desk
- Email: help@pawsey.org.au

pawsey

Thank you for attending!

Please complete this short survey:

https://www.surveymonkey.com/r/Y3YFYHQ