OpenFlightHPC-compute Documentation

OpenFlightHPC

Apr 08, 2020

Overview

This site contains the documentation to get you started with using an OpenFlightHPC Compute research environment. It provides details for accessing the research environment, exploring the various features of the research environment and examples of common workflows with the research environment.

CHAPTER 1

Acknowledgements

We recognise the respect the trademarks of all third-party providers referenced in this documentation. Please see the respective EULAs for software packages used in configuring your own environment based on this knowledgebase.

1.1 License

CHAPTER 2

Table of Contents

2.1 What is OpenFlightHPC Compute?

OpenFlight Compute is the software environment designed to help researchers and scientists run their own high-performance compute research environment quickly and easily. The basic structure provided for users is as follows:

- One gateway node, plus a configurable number of compute nodes
- An Enterprise Linux operating system
- A shared filesystem, mounted across all nodes
- A batch job scheduler
- Access to various libraries of software applications

OpenFlight Compute is designed to get researchers started with HPC as quickly as possible, providing a pre-configured environment which is ready for work immediately. The research environment you build is personal to you - users have root-access to the environment, and can setup and configure the system to their needs.

Your research environment is designed to be **ephemeral** - i.e. you run it for as long as you need it, then shut it down. Although there is no built-in time limit for OpenFlight Compute research environments, the most effective way of sharing compute resources in the cloud is to book them out only when you need them. Contrary to popular belief, you can achieve huge cost savings over purchasing server hardware if you learn to work effectively in this way.

2.1.1 Who is it for?

OpenFlight Compute is designed for use by end-users - that's the scientists, researchers, engineers and software developers who actually run compute workloads and process data. This documentation is designed to help these people to get the best out this environment, without needing assistance from teams of IT professionals. Flight provides tools which enable users to service themselves - it's very configurable, and can be expanded by individual users to deliver a scalable platform for computational workloads.

2.1.2 What doesn't it do?

An important part of having ultimate power to control your environment is taking responsibility for it. While no one is going to tell you how you should configure your research environment, you need to remember good security practice, and look after both your personal and research data. OpenFlight Compute provides you with a personal, single-user research environment - please use responsibility. OpenFlight Compute is not indended as a replacement for your national super-computer centre.

If you're running OpenFlight Compute on AWS, then there are some great tutorials written by the Amazon team on how to secure your environment. Just because you're running on public cloud, doesn't mean that your research environment is any less secure than a research environment running in your basement. Start at the AWS Security Pages, and talk to a security expert if you're still unsure.

2.2 Prerequisites

You're going to need access to some computers. If you already have access to cloud resources, then great - you're ready to go. There are some specific requirements depending on your platform type, which are discussed in the relevant chapters of this guide. If you don't have access to anything yet then that's fine too - just sign up for an AWS account now and you'll have all the access you need.

You'll need a client device as well - something to log into your research environment from. The requirements on client devices are fairly minimal, but you'll need:

- **A computer with a screen and a keyboard**; you can probably access on a tablet or smartphone too, if your typing is good enough
- **A relatively modern web-browser**; Apple Safari, Google Chrome, Microsoft Edge, Mozilla Firefox, and pretty much anything else that was written/updated in the last couple of years should all be fine.
- **An SSH client**; this comes built-in for Linux and Mac based machines, but you'll need to install one for Windows clients and some tablets.
- **Internet access**; it seems dumb to list this as a requirement for running HPC on cloud resources, but a surprising number of sites actually limit outbound connectivity to the Internet. You'll need to be able to access the Internet from your client device.
- *Optionally*; **A graphical data transfer tool**; you don't actually need one of these, but they can really help new users get started more quickly.

It's worth checking for centrally-managed client systems that you can install the software that you need - some research sites don't allow users to install new software. Here are some recommendations of software that you can use on client machines; this is far from a complete list, but should help you get started:

- **SSH client:**
  - Use the built-in `ssh` client for Mac and Linux
  - For Windows, try Putty or SmaTTY
- **Web-browser:**
  - Use the built-in Safari browser on Macs
  - For Linux and Windows, install Firefox or Chrome
- **VNC (Graphical desktop client):**
  - Use the built-in VNC client on Macs
  - For Linux, install "vncviewer" package, or install RealVNC viewer

  - For Windows, install TurboVNC

- **Graphical file-transfer tools:**
  - For Macs and Linux, install Cyberduck or Filezilla
  - For Windows, try WinSCP, Cyberduck or Filezilla

We've tried to make recommendations for open-source and/or free software client software here - as ever, please read and obey the licensing terms, and try to contribute to the supporting projects either financially, or by referencing them in your research publications.

2.3 Where can I get help?

This documentation is designed to walk users through the first stages of creating their research environments, and getting started in the environment. Capable users with some experience can be up and running in a handful of minutes - don't panic if it takes you a little more time, especially if you've not used Linux or HPC research environments before. Firstly - don't worry that you might break something complicated and expensive; one of the joys of having your own personal environment is that no one will tell you that you're doing it wrong, and nothing is at risk of being broken, aside from the data and work you've done yourself in the environment.

We encourage new users to run through a few tutorials in this documentation - even if you have plenty of HPC experience. Technology moves forward all the time and new features are constantly popping up that could save you effort in future. If you do run into problems, try replicating the steps you went through to get where you are - sometimes a typo in a command early-on in your workflow might not cause any errors until right at the end of your work. It can help to work collaboratively with other researchers running similar jobs - not only are two sets of eyes better than one, you'll both get something out of working together to achieve a shared goal.

There is a community site for supporting the OpenFlight software - available online here. This website is designed to help users share their experiences of running Flight research environments, report any bugs with the software, and share knowledge to help everyone work more effectively. There is no payment required for using this service, except for the general requirement to be nice to each other - if you find the site useful, then please pay the favour back by helping another user with their problem.

The OpenFlight community support site is a great resource for helping with HPC research environment usage, but for software application support you're going to need to contact the developers of the packages themselves. Remember that many of these software products are open-source and you've paid no fee to use them - try to make your bug-reports and enhancement requests as helpful and friendly as possible to the application developers. They've done you a great service by making their software available for you to use - please be respectful of their time and effort if you need to contact them, and remember to credit their software in your research publications.

2.4 What is Flight User Suite?

The Flight User Suite is a collection of environment tools that provide users with easy and intuitive ways to manage the software and desktop sessions in a research envrionment. The purpose of these tools is to get researchers started with HPC as quickly as possible without needing to worry about their environment, leaving them to do what they do best - research!

The tools are non-intrusive to the research environment, defaulting to a "deactivated" state. Leaving admins and users free to configure and utilise their systems how they want.

Flight User Suite is made up of the following tools:

- Runway: Flight Runway provides a self-contained Ruby environment and an entrypoint for accessing the other flight tools

- Env: Flight Env provides access to, and management of, various software managers to ensure access to a wide variety of HPC applications
- Desktop: Flight Desktop provides an intuitive tool for launching VNC-ready virtual desktops of many different desktop environments (gnome, xterm, kde, etc)
- Starter: Flight Starter provides profile scripts for integrating the user suite into the shell environment

2.5 Installing Flight User Suite

The OpenFlight project packages tools as RPMs and hosts them in a yum repository that can be quickly installed via a release RPM.

2.5.1 Adding the OpenFlight Yum Repositories

CentOS 7

- Install the OpenFlight release RPM:

```
[flight@gateway1 ~]$ yum install https://repo.openflighthpc.org/openflight/centos/
↪7/x86_64/openflighthpc-release-3-1.noarch.rpm
```

- Rebuild the yum cache:

```
[flight@gateway1 ~]$ yum makecache
```

CentOS 8

- Install the OpenFlight release RPM:

```
[flight@gateway1 ~]$ dnf install https://repo.openflighthpc.org/openflight/centos/
↪8/x86_64/openflighthpc-release-3-1.noarch.rpm
```

- Rebuild the yum cache:

```
[flight@gateway1 ~]$ dnf makecache
```

Ubuntu 18.04

Coming Soon...

Ubuntu 20.04

Coming Soon...

Now the OpenFlight repositories are installed. There are 3 repositories available - production (enabled by default), dev (providing development releases of tools) and vault (access to old, unsupported versions and retired tools).

2.5.2 Installation Method 1: Quick

The quickest and simplest way to get up and running with the user suite is to simply install the group package for the tools. This will ensure that compatible versions of all the tools are installed.

CentOS 7

- Install the user suite RPM:

```
[flight@gateway1 ~]$ yum install flight-user-suite
```

CentOS 8

- Install the user suite RPM:

```
[flight@gateway1 ~]$ dnf install flight-user-suite
```

Ubuntu 18.04

Coming Soon...

Ubuntu 20.04

Coming Soon...

Note: After installation, either reboot your system or logout and back in again to expose the `flight` command to the shell

---

2.5.3 Installation Method 2: Slightly Less Quick

Each tool in the user suite is also available through the repositories and can be installed one at a time.

CentOS 7

- Install the Flight Runway RPM:

```
[flight@gateway1 ~]$ yum install flight-runway
```

- Install Flight Env RPM:

```
[flight@gateway1 ~]$ yum install flight-env
```

- Install Flight Desktop RPM:

```
[flight@gateway1 ~]$ yum install flight-desktop
```

- Install Flight Starter RPM:

```
[flight@gateway1 ~]$ yum install flight-starter
```

CentOS 8

- Install the Flight Runway RPM:

```
[flight@gateway1 ~]$ dnf install flight-runway
```

- Install Flight Env RPM:

```
[flight@gateway1 ~]$ dnf install flight-env
```

- Install Flight Desktop RPM:

```
[flight@gateway1 ~]$ dnf install flight-desktop
```

- Install Flight Starter RPM:

```
[flight@gateway1 ~]$ dnf install flight-starter
```

Ubuntu 18.04

Coming Soon...

Ubuntu 20.04

Coming Soon...

Note: After installation, either reboot your system or logout and back in again to expose the `flight` command to the shell

2.5.4 Installation Method 3: Manual

For those who wish to have more control over their installation, all of the Flight User Suite tools have manual installation instructions in the READMEs on GitHub.

- Flight Runway - https://github.com/openflighthpc/flight-runway#manual-installation
- Flight Env - https://github.com/openflighthpc/flight-env#installation
- Flight Desktop - https://github.com/openflighthpc/flight-desktop#from-source
- Flight Starter - https://github.com/openflighthpc/flight-starter#installation

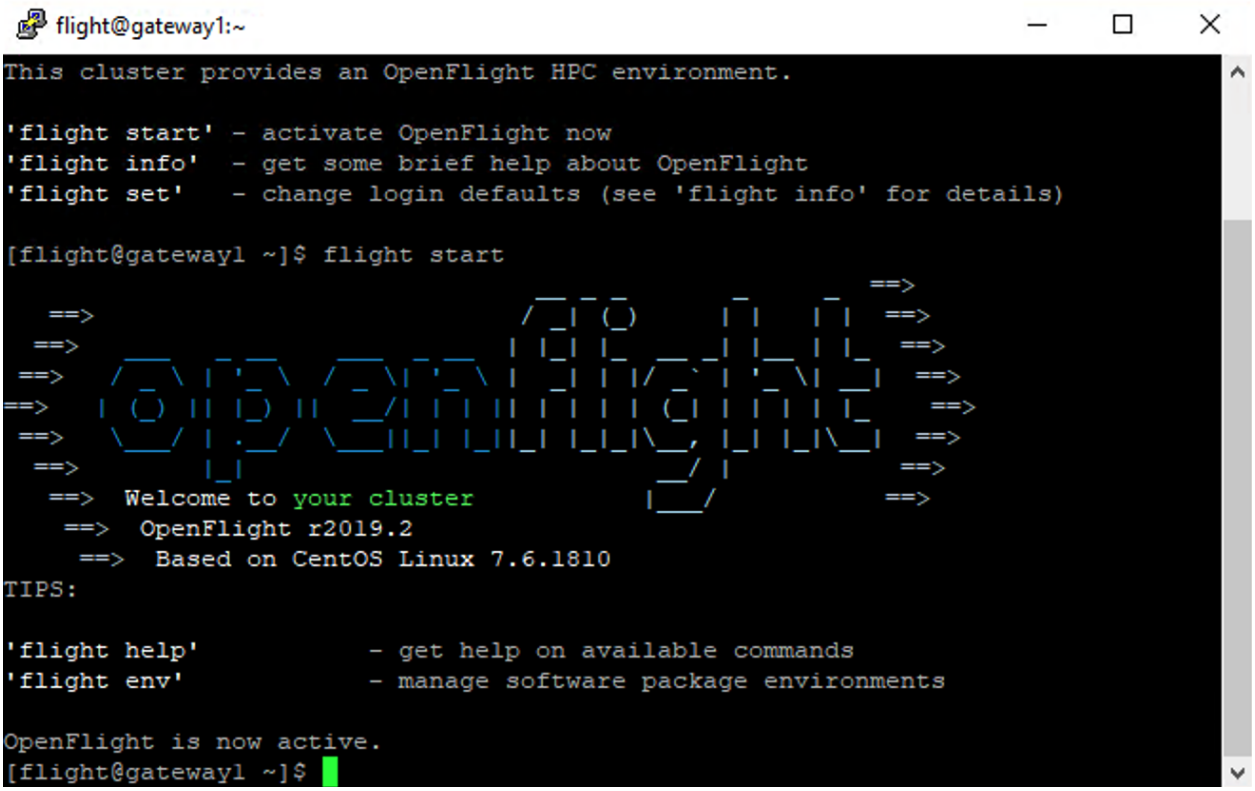
2.6 Flight System

2.6.1 Activating the Flight System

The Flight User Suite sits unobtrusively on the research environment with the *flight* command serving as an entrypoint to the various system commands.

This provides some quick tips to activating the system and finding out more about the flight system (`flight info`).

To load the system, simply run `flight start` (which, when first run, will generate some login keys for allowing passwordless login to compute nodes)



Tip: The flight system can be set to automatically start on login for the user by running `flight set always on`

2.6.2 Customising the Environment

Setting the Research Environment Name

To identify the research environment that's being worked on, a cluster name can be specified and displayed in the prompt when the Flight System is active.

A cluster name can be specified as follows:

```
[flight@gateway1 ~]$ flight config set cluster.name scooby
```

2.7 Flight Environment

2.7.1 Viewing Available Ecosystems

Various *Package Ecosystems* are available for managing software on your Flight research environment. These can be viewed by using the `env` subcommand:

```
[flight@gateway1 (scooby) ~]$ flight env avail
```

2.7.2 Creating a Local Ecosystem

A local ecosystem is only available to the user that creates it. All of the packages and libraries are installed to the users home directory.

To install a package ecosystem, use the create command as follows (replacing gridware with your desired package ecosystem):

```
[flight@gateway1 (scooby) ~]$ flight env create gridware
```

Once a package ecosystem has been installed, it needs to be activated for the session to be able to manage software with it:

```
[flight@gateway1 (scooby) ~]$ flight env activate gridware
<gridware> [flight@gateway1 (scooby) ~]$
```

Tip: Your preferred software ecosystem can be set to automatically activate for your user within the flight system by running `flight env set-default gridware`, replacing gridware with your chosen software ecosystem

2.7.3 Creating a Global Ecosystem

A global ecosystem is available to all users on the system. All of the packages and libraries are installed to a shared storage directory. The global directories can be configured in `/opt/flight/opt/flight-env/etc/config.yml` with the `global_depot_path:` and `global_build_cache_path` keys.

Note: The user requires suitable write permissions to the configured global depot paths in order to be able to create a global ecosystem

To install a global package ecosystem, use the create command with the global option flag:

```
[root@gateway1 (scooby) ~]$ flight env create -g gridware
```

Once the global ecosystem has been installed, it needs to be activated for the session to be able to monitor software with it:

```
[root@gateway1 (scooby) ~]$ flight env activate gridware@global
<gridware@global> [flight@gateway1 (scooby) ~]$
```

2.7.4 Custom Ecosystem Names

When installing an ecosystem, an custom alias can be added by appending `@mycustomname` to the end of creation command. For example, to create a local gridware installation with the alias `test`:

```
[flight@gateway1 (scooby) ~]$ flight env create gridware@test
```

To activate this environment, the alias will need to be specified in the activation command:

```
[flight@gateway1 (scooby) ~]$ flight env activate gridware@test
<gridware@test> [flight@gateway1 (scooby) ~]$
```

2.8 Flight Desktop

Your OpenFlight Compute gateway node can run graphical desktop sessions to support users who want to run interactive applications across the research environment. The system can support a number of different sessions simultaneously, and allow multiple remote participants to connect to the same session to support training and collaborative activities.

2.8.1 Install Flight Desktop Types

Your OpenFlight Compute research environment supports many types of graphical session designed to provide interactive applications directly to users. To view the available types of session, use the command `flight desktop avail`:

```
[flight@gateway1 (scooby) ~]$ flight desktop avail
```

Name	Summary	State
chrome	Full-screen Google Chrome browser session. > https://www.google.com/chrome/	Unverified
gnome	GNOME v3, a free and open-source desktop environment for Unix-like operating systems. > https://www.gnome.org/	Unverified
kde	KDE Plasma Desktop (KDE 4). Plasma is KDE's desktop environment. Simple by default, powerful when needed. > https://kde.org/	Unverified
terminal	Preconfigured terminal for Flight HPC environments.	Unverified
xfce	Xfce is a lightweight desktop environment for UNIX-like operating systems. It aims to be fast and low on system resources, while still being visually appealing and user friendly. > https://xfce.org/	Unverified

(continues on next page)

(continued from previous page)

```
│          │                                                                      ␣
↪                              │            │
│ xterm    │ Minimal desktop environment with an xterm terminal window.           ␣
↪                              │ Unverified │
│          │                                                                      ␣
↪                              │            │
│          │  > https://invisible-island.net/xterm/xterm.html                     ␣
↪                              │            │
│          │                                                                      ␣
↪                              │            │
└
```

Application types that are `unverified` need to be prepared before they can be started. To prepare a new session type, use the command `flight desktop prepare <type>` (preparing will automatically install any required application and support files, if these dependencies have been installed manually then a desktop session can be checked for verfication with `flight desktop verify <type>`). Once enabled, users can start a new session using the command `flight desktop start <type>`.

Note: The `prepare` command is only available to the `root` user as it requires installation of packages

Note: Preparing a new session type only enables it for the machine that you run the command from, any other nodes will need to have the type enabled too.

2.8.2 Launch a Desktop Session

Users can launch a new session by using the `flight desktop start gnome` command. After launching the desktop, a message will be pri nted with connection details to access the new session:

```
Starting a 'gnome' desktop session:

   >  Starting session

A 'gnome' desktop session has been started.

== Session details ==

  Identity: d33096d5-b50c-49dd-a59e-d5aebc7940ac
      Type: gnome
   Host IP: 51.104.217.65
  Hostname: gateway1
      Port: 5902
   Display: :2
  Password: L9Uysvi3

This desktop session is not directly accessible from outside of your
cluster as it is running on a machine that only provides internal
cluster access.  In order to access your desktop session you will need
to perform port forwarding using 'ssh'.

Refer to 'flight desktop show d33096d5' for more details.
```

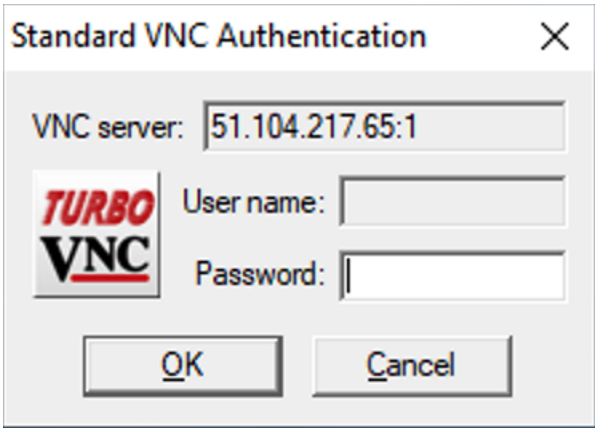
(continues on next page)

(continued from previous page)

```
If prompted, you should supply the following password: L9Uysvi3
```

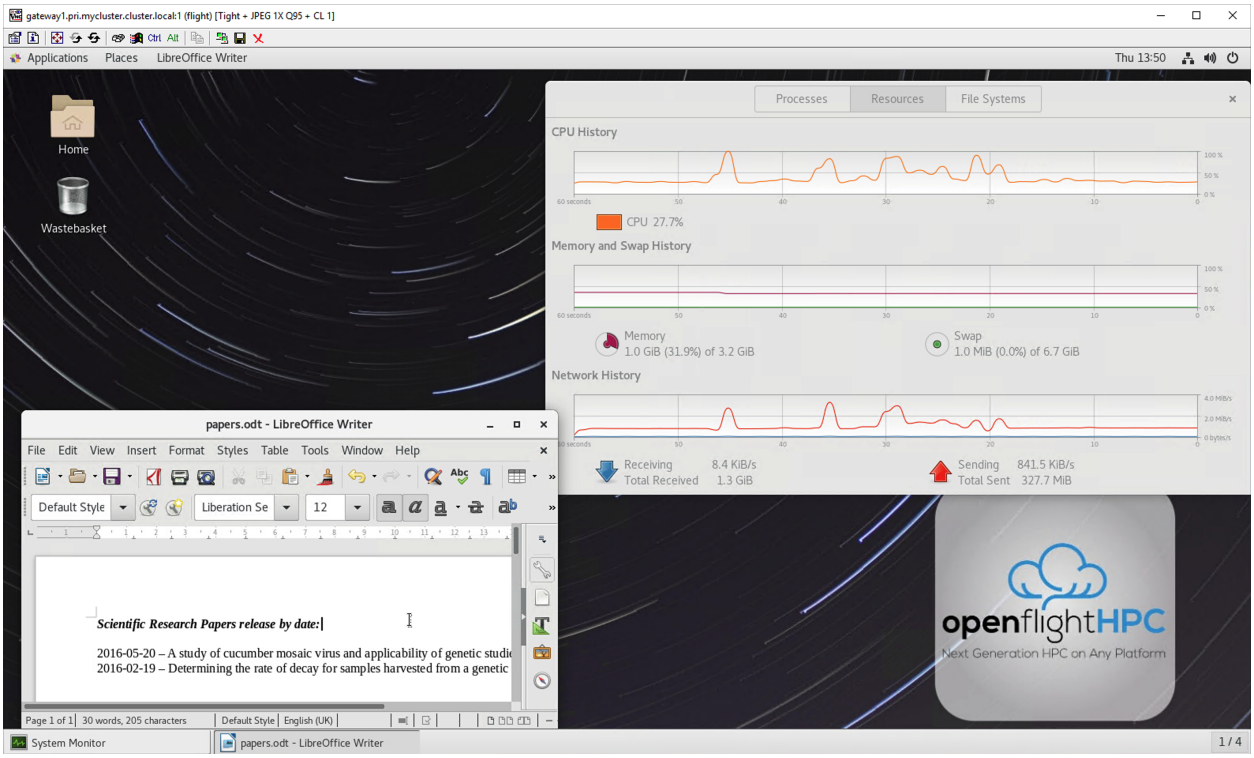
Users need a VNC client to connect to the graphical desktop session - for a list of tested clients, see *Prerequisites*.

Users with Mac clients can use the URL provided in the command output to connect to the session; e.g. from the above example, simply enter `vnc://flight:L9Uysvi3@52.151.119.86:5902` into the Safari address bar. Linux and Windows users should enter the IP address and port number shown into their VNC client in the format `IP:port`. For example - for the output above, Linux and Windows client users would enter `52.151.119.86:5902` into their VNC client:



A one-time randomized password is automatically generated automatically by OpenFlight Compute when a new session is started. Linux and Windows users may be prompted to enter this password when they connect to the desktop session.

Once connected to the graphical desktop, users can use the session as they would a local Linux machine:



2.8.3 Resizing the desktop to fit your screen

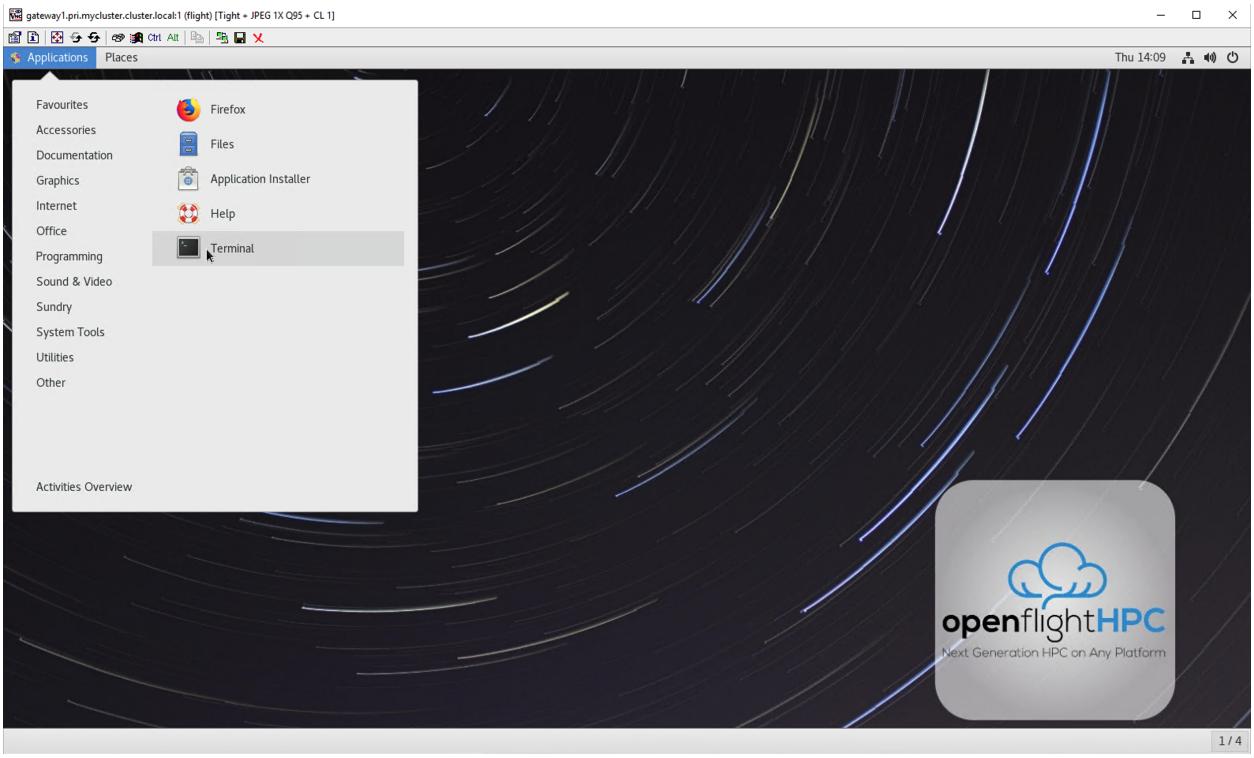
Specifying a size with the flight desktop tool

When launching a graphical desktop session using the `flight desktop` utility, a session resolution can be specified using the `--geometry <size>` option. For example, to launch a `gnome` desktop session with a resolution of 1920x1080 pixels, use the command:

```
[flight@gateway1(scooby) ~]$ flight desktop start --geometry 1920x1080 gnome
```

By default, your graphical desktop session will launch with a compatibility resolution of 1024x768. Users can resize the desktop to fit their screens using the Linux `xrandr` command, run from within the graphical desktop session.

To view the available screen resolutions, start a terminal session on your graphical desktop by navigating to the `Applications` menu in the top left-hand corner of the screen, then selecting the `Terminal` under the `System tools` menu.



The `xrandr` command will display a list of available resolutions supported by your desktop:

```
[flight@gateway1(scooby) ~]$ xrandr
Screen 0: minimum 32 x 32, current 1024 x 768, maximum 32768 x 32768
VNC-0 connected primary 1024x768+0+0 0mm x 0mm
   1920x1200     60.00
   1920x1080     60.00
   1600x1200     60.00
   1680x1050     60.00
   1400x1050     60.00
   1360x768      60.00
   1280x1024     60.00
   1280x960      60.00
   1280x800      60.00
```

(continues on next page)

(continued from previous page)

```
   1280x720      60.00
   1024x768      60.00*
   800x600       60.00
   640x480       60.00
```

To set a new resolution, run the `xrandr` command again with the `-s <resolution>` argument;

- e.g. to change to 1280x1024, enter the command `xrandr -s 1280x1024`

Your graphical desktop session will automatically resize to the new resolution requested. Use your local VNC client application to adjust the compression ratio, colour depth and frame-rate sessions in order to achieve the best user-experience for the desktop session.

2.8.4 Viewing and terminating running sessions

Users can view a list of the currently running sessions by using the command `flight desktop list`. One standard Flight Compute login node supports up to 10 sessions running at the same time.

```
[flight@gateway1 (scooby) ~]$ flight desktop list
┌──────────┬──────────┬──────────┬──────────────┬──────────────┬──────────┬──────
│ Identity │ Type     │ Host name │ IP address   │ Display (Port) │ Password │ State␣
↪ │
├──────────┼──────────┼──────────┼──────────────┼──────────────┼──────────┼──────
│ 6200f57c │ terminal │ gateway1  │ 51.104.217.65 │ :2 (5902)      │ KbnGqk0L │␣
↪Active │
│ 70e75a99 │ chrome   │ gateway1  │ 51.104.217.65 │ :4 (5904)      │ UjlrnN9f │␣
↪Active │
│ 809b9466 │ gnome    │ gateway1  │ 51.104.217.65 │ :1 (5901)      │ uxAMZfB7 │␣
↪Active │
│ a43059c5 │ gnome    │ gateway1  │ 51.104.217.65 │ :3 (5903)      │ Ofd1qVdN │␣
↪Active │
└──────────┴──────────┴──────────┴──────────────┴──────────────┴──────────┴──────
```

To display connection information for an existing session, use the command `flight desktop show <session-ID>`. This command allows users to review the IP-address, port number and one-time password settings for an existing session.

```
[flight@gateway1 (scooby) ~]$ flight desktop show 6200f57c

== Session details ==

  Identity: 6200f57c-ead7-45d5-901d-0b1f9a1d2dad
      Type: terminal
   Host IP: 51.104.217.65
  Hostname: gateway1
      Port: 5902
   Display: :2
  Password: KbnGqk0L

This desktop session is accessible from the public internet. However,
please be aware that desktop sessions accessed over the public
internet are not secure and steps should be taken to secure the link.

We highly recommend that you access your desktop session using 'ssh'
port forwarding:
```

(continues on next page)

(continued from previous page)

```
  ssh -L 5901:localhost:5902 flight@51.104.217.65

Once the ssh connection has been established, depending on your
client, you can connect to the session using one of:

  vnc://flight:KbnGqk0L@localhost:5901
  localhost:5901
  localhost:1

If, when connecting, you receive a warning as follows, try again with
a different port number, e.g. 5902, 5903 etc.:

  channel_setup_fwd_listener_tcpip: cannot listen to port: 5901

If prompted, you should supply the following password: KbnGqk0L
```

Users can terminate a running session by ending their graphical application (e.g. by logging out of a Gnome session, or exiting a terminal session), or by using the `flight desktop kill <session-ID>` command. A terminated session will be immediately stopped, disconnecting any users.

2.8.5 Securing your graphical desktop session

As the VNC protocol does not natively provide support for security protocols such as SSL, you may wish to take steps to secure access to your VNC sessions.

Several third party tools exist to help you secure your VNC connections. One option is *ssvnc*, available here.

Alternatively, you could use an SSH tunnel to access your session. Refer to online guides for setup instructions.

2.9 Package Ecosystems

The Flight Env command provides streamlined installation of many popular software management ecosystems. These can be seen below along with the features they provide.

	conda	easybuild	gridware	modules	singularity	spack
Dependency Management	y	y	y	n	y	y
Compilation	y	y	n	n	n	y
Preconfigured Binaries	n	y	y	n	y	*
Multiple Versions	*	y	y	y	y	y

More information on the features:

- **Dependency Management** - The ecosystem automatically resolves and installs dependencies
- **Compilation** - The ecosystem builds software from source for optimised functionality on the system (Note: these sorts of installations will take a long time)
- **Preconfigured Binaries** - The ecosystem provides binaries of the available software for quicker installation to get applications installed quickly
- **Multiple Versions** - The ecosystem allows for installation of multiple versions of software which can be toggled between

⋆ is for partial support. Technically these features are supported in certain use cases and with additional work. This is not considered full support as multiple versions may require undocumented or uncommon use-cases of the tool.

2.9.1 Conda

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

2.9.2 Easybuild

EasyBuild is a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way.

2.9.3 Gridware

Gridware provides pre-compiled binaries for many different scientific computing programs which can be installed quickly and loaded into the environment via modules.

2.9.4 Modules

Modules provides simple environment management. Tools and software to be used with modules will be compiled or installed outside of the modules environment.

2.9.5 Singularity

Singularity is high-performance container technology specifically designed to enhance Enterprise Performance Computing by building containers that support HPC, analytics, artificial intelligence, machine learning, and deep learning to provide "intelligence anywhere."

2.9.6 Spack

Spack is a package manager for supercomputers, Linux, and macOS. It makes installing scientific software easy. With Spack, you can build a package with multiple versions, configurations, platforms, and compilers, and all of these builds can coexist on the same machine.

2.10 Conda: Usage Example

2.10.1 Creating and Using Ecosystem

Flight Env provides quick setup methods to create a conda software ecosystem.

To install and use conda:

- *Activate the flight system*
- Create the conda installation for the user:

```
[flight@gateway1 ~]$ flight env create conda
Creating environment conda@default
   >  Verifying prerequisites
   >  Fetching prerequisite (miniconda)
   >  Creating environment (conda@default)
Environment conda@default has been created
```

- Activate the conda ecosystem:

```
[flight@gateway1 ~]$ flight env activate conda
(base) <conda> [flight@gateway1 ~]$
```

- Check that conda can be run:

```
(base) <conda> [flight@gateway1 ~]$ conda --version
conda 4.7.10
```

2.10.2 Installing and Running Perl

An example workflow using perl is demonstrated below.

- View available versions:

```
(base) <conda> [flight@gateway1 ~]$ conda search perl
Loading channels: done
# Name                       Version           Build  Channel
perl                          5.26.0      hae598fd_0  pkgs/main
perl                          5.26.2      h14c3975_0  pkgs/main
```

- Install specific version:

```
(base) <conda> [flight@gateway1 ~]$ conda install perl=5.26.2
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/flight/.local/share/flight/env/conda+default

  added / updated specs:
    - perl=5.26.2


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2019.5.15  |                1         134 KB
    certifi-2019.6.16          |           py37_1         156 KB
    conda-4.7.11               |           py37_0         3.0 MB
    perl-5.26.2                |       h14c3975_0        10.5 MB
    ------------------------------------------------------------
                                           Total:        13.7 MB

The following NEW packages will be INSTALLED:
```

(continues on next page)

(continued from previous page)

```
  perl               pkgs/main/linux-64::perl-5.26.2-h14c3975_0

The following packages will be UPDATED:

  ca-certificates                                2019.5.15-0 --> 2019.5.15-1
  certifi                                    2019.6.16-py37_0 --> 2019.6.16-py37_1
  conda                                         4.7.10-py37_0 --> 4.7.11-py37_0


Proceed ([y]/n)? y


Downloading and Extracting Packages
conda-4.7.11          | 3.0 MB    | ############################################
↪############################################ | 100%
certifi-2019.6.16     | 156 KB    | ############################################
↪############################################ | 100%
perl-5.26.2           | 10.5 MB   | ############################################
↪############################################ | 100%
ca-certificates-2019 | 134 KB    | ############################################
↪############################################ | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

- Check installation location:

```
(base) <conda> [flight@gateway1 ~]$ which perl
~/.local/share/flight/env/conda+default/bin/perl
```

- Install perl library (this may prompt for initial `cpan` configuration, once configuration is complete then the library will be installed):

```
(base) <conda> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal null logger. Install Log::Log4perl for logging messages
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Mon, 09 Sep 2019 15:17:03 GMT
Running install for module 'File::Slurp'
<-- snip -->
```

- Check installation worked:

```
(base) <conda> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal null logger. Install Log::Log4perl for logging messages
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Mon, 09 Sep 2019 15:17:03 GMT
File::Slurp is up to date (9999.27).
```

2.11 Easybuild: Usage Example

2.11.1 Creating and Using Ecosystem

Flight Env provides quick setup methods to create an easybuild software ecosystem.

To install and use easybuild:

- *Activate the flight system*
- Create the easybuild installation for the user:

```
[flight@gateway1 ~]$ flight env create easybuild
Creating environment easybuild@default
   >  Verifying prerequisites
   >  Fetching prerequisite (lua)
   >  Extracting prerequisite (lua)
   >  Building prerequisite (lua)
   >  Installing prerequisite (lua)
   >  Fetching prerequisite (tcl)
   >  Extracting prerequisite (tcl)
   >  Building prerequisite (tcl)
   >  Installing prerequisite (tcl)
   >  Fetching prerequisite (lmod)
   >  Extracting prerequisite (lmod)
   >  Configuring prerequisite (lmod)
   >  Installing prerequisite (lmod)
   >  Fetching prerequisite (easybuild)
   >  Bootstrapping EasyBuild environment (easybuild@default)
Environment easybuild@default has been created
```

- Activate the easybuild ecosystem:

```
[flight@gateway1 ~]$ flight env activate easybuild
<easybuild> [flight@gateway1 ~]$
```

- Check that easybuild can be run:

```
<easybuild> [flight@gateway1 ~]$ module load EasyBuild
<easybuild> [flight@gateway1 ~]$ eb --version
This is EasyBuild 3.9.4 (framework: 3.9.4, easyblocks: 3.9.4) on host gateway1.
↪pri.basic.cluster.local.
```

2.11.2 Installing and Running Perl

An example workflow using perl is demonstrated below.

- View available versions:

```
<easybuild> [flight@gateway1 ~]$ eb -S perl
CFGS1=/home/flight/.local/share/flight/env/easybuild+default/software/EasyBuild/3.
↪9.4/lib/python2.7/site-packages/easybuild_easyconfigs-3.9.4-py2.7.egg/easybuild/
↪easyconfigs
 * $CFGS1/a/annovar/annovar-2016Feb01-foss-2016a-Perl-5.22.1.eb
 * $CFGS1/b/Bio-DB-HTS/Bio-DB-HTS-2.11-foss-2017b-Perl-5.26.0.eb
 * $CFGS1/b/Bio-DB-HTS/Bio-DB-HTS-2.11-foss-2018b-Perl-5.28.0.eb
 * $CFGS1/b/Bio-DB-HTS/Bio-DB-HTS-2.11-intel-2017b-Perl-5.26.0.eb
<-- snip -->
 * $CFGS1/p/Perl/Perl-5.26.1-GCCcore-6.4.0.eb
 * $CFGS1/p/Perl/Perl-5.26.1-foss-2018a-bare.eb
 * $CFGS1/p/Perl/Perl-5.26.1-foss-2018a.eb
 * $CFGS1/p/Perl/Perl-5.28.0-GCCcore-7.3.0.eb
 * $CFGS1/p/Perl/Perl-5.28.1-GCCcore-8.2.0.eb
<-- snip -->
 * $CFGS1/y/YAML-Syck/YAML-Syck-1.27-goolf-1.4.10-Perl-5.16.3.eb
```

(continues on next page)

(continued from previous page)

```
 * $CFGS1/y/YAML-Syck/YAML-Syck-1.27-ictce-5.3.0-Perl-5.16.3.eb

Note: 9 matching archived easyconfig(s) found, use --consider-archived-
↪easyconfigs to see them
```

- Install specific version:

```
<easybuild> [flight@gateway1 ~]$ eb Perl-5.28.1-GCCcore-8.2.0.eb --robot
== temporary log file in case of crash /tmp/eb-MdohD2/easybuild-J6tjhZ.log
== resolving dependencies ...
== processing EasyBuild easyconfig /home/flight/.local/share/flight/env/
↪easybuild+default/software/EasyBuild/3.9.4/lib/python2.7/site-packages/
↪easybuild_easyconfigs-3.9.4-py2.7.egg/easybuild/easyconfigs/m/M4/M4-1.4.18.eb
== building and installing M4/1.4.18...
== fetching files...
== creating build dir, resetting environment...
== unpacking...
== patching...
== preparing...
== configuring...
== building...
<-- snip -->
```

- Check installation location:

```
<easybuild> [flight@gateway1 ~]$ which perl
```

- Install perl library (this may prompt for initial `cpan` configuration, once configuration is complete then the library will be installed):

```
<easybuild> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal null logger. Install Log::Log4perl for logging messages
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Mon, 09 Sep 2019 15:47:03 GMT
Running install for module 'File::Slurp'
<-- snip -->
```

- Check installation worked:

```
<easybuild> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal null logger. Install Log::Log4perl for logging messages
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Mon, 09 Sep 2019 15:47:03 GMT
File::Slurp is up to date (9999.27).
```

2.12 Gridware: Usage Example

2.12.1 Creating and Using Ecosystem

Flight Env provides quick setup methods to create a gridware software ecosystem.

To install and use gridware:

- *Activate the flight system*

- Create the gridware installation for the user:

```
[flight@gateway1 ~]$ flight env create gridware
Creating environment gridware@default
   >  Verifying prerequisites
   >  Fetching prerequisite (modules)
   >  Extracting prerequisite (modules)
   >  Building prerequisite (modules)
   >  Installing prerequisite (modules)
   >  Fetching prerequisite (gridware)
   >  Extracting prerequisite (gridware)
   >  Installing prerequisite (gridware)
   >  Configuring repo: main
   >  Configuring repo: volatile
   >  Creating environment (gridware@default)
Environment gridware@default has been created
```

- Activate the gridware ecosystem:

```
[flight@gateway1 ~]$ flight env activate gridware
<gridware> [flight@gateway1 ~]$
```

- Check that gridware can be run:

```
<gridware> [flight@gateway1 ~]$ gridware --version
gridware 1.5.1
```

2.12.2 Installing and Running Perl

An example workflow using perl is demonstrated below.

- View available versions:

```
<gridware> [flight@gateway1 ~]$ gridware search
2 repositories need to update ...
Updating repository: main
          Update ... OK (At: 55916fd)
Updating repository: volatile
          Update ... OK (At: 61cc544)
main/apps/bismark/0.16.3                  main/apps/bismark/0.17.0
 ↪  main/apps/cegma/2.5.0
main/apps/clusterflow/0.6.0-20170502      main/apps/clusterflow/0.6.0-20170530
 ↪  main/apps/forks/0.36
main/apps/ihrsr/1.5                       main/apps/perl/5.20.2
 ↪  main/apps/python/2.7.8
<-- snip -->
volatile/apps/perl/5.16.1                 volatile/apps/perl/5.16.3
 ↪  volatile/apps/perl/5.18.0
volatile/apps/perl/5.20.2                 volatile/apps/python/2.7.3
 ↪  volatile/apps/python/2.7.5
<-- snip -->
```

- Install specific version:

```
<gridware> [flight@gateway1 ~]$ gridware install main/apps/perl/5.20.2
Preparing to install main/apps/perl/5.20.2
Installing main/apps/perl/5.20.2
```

(continues on next page)

(continued from previous page)

```
 > Preparing package sources
        Download --> perl-5.20.2.tar.gz ... OK
          Verify --> perl-5.20.2.tar.gz ... OK
        Packaged --> CPAN-Config.pm     ... OK
        Packaged --> CPAN-MyConfig.pm   ... OK

 > Preparing for installation
           Mkdir ... OK (/home/flight/.cache/flight/env/build/gridware/src/apps/
↪perl/5.20.2/gcc-4.8.5)
         Extract ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK (/home/flight/.local/share/flight/env/gridware+default/
↪depots/23cd1570/el7/pkg/apps/perl/5.20.2/gcc-4.8.5)
         Install ... OK
          Module ... OK

Installation complete.
```

- Check installation location:

```
<gridware> [flight@gateway1 ~]$ module load apps/perl
apps/perl/5.20.2/gcc-4.8.5
 | -- libs/gcc/system
 |    * --> OK
 |
 OK
<gridware> [flight@gateway1 ~]$ which perl
~/.local/share/flight/env/gridware+default/depots/23cd1570/el7/pkg/apps/perl/5.20.
↪2/gcc-4.8.5/bin/perl
```

- Install perl library (this may prompt for initial `cpan` configuration, once configuration is complete then the library will be installed):

```
<gridware> [flight@gateway1 ~]$ cpan File::Slurp
Reading '/home/flight/gridware/share/perl/5.20.2/cpan/Metadata'
  Database was generated on Tue, 10 Sep 2019 01:17:03 GMT
Running install for module 'File::Slurp'
<-- snip -->
Appending installation info to /home/flight/gridware/share/perl/5.20.2/lib/5.20.2/
↪x86_64-linux/perllocal.pod
  CAPOEIRAB/File-Slurp-9999.27.tar.gz
  /usr/bin/make install INSTALLMAN3DIR=/home/flight/gridware/share/perl/5.20.2/
↪man/man3 INSTALLMAN1DIR=/home/flight/gridware/share/perl/5.20.2/man/man1 -- O
```

- Check installation worked:

```
<gridware> [flight@gateway1 ~]$ cpan File::Slurp
Reading '/home/flight/gridware/share/perl/5.20.2/cpan/Metadata'
  Database was generated on Tue, 10 Sep 2019 01:17:03 GMT
File::Slurp is up to date (9999.27).
```

2.13 Modules: Usage Example

2.13.1 Creating and Using Ecosystem

Flight Env provides quick setup methods to create a modules software ecosystem.

To install and use modules:

- *Activate the flight system*
- Create the modules installation for the user:

```
[flight@gateway1 ~]$ flight env create modules
Creating environment modules@default
   >  Verifying prerequisites
   >  Fetching prerequisite (modules)
   >  Extracting prerequisite (modules)
   >  Building prerequisite (modules)
   >  Installing prerequisite (modules)
   >  Creating environment (modules@default)
Environment modules@default has been created
```

- Activate the modules ecosystem:

```
[flight@gateway1 ~]$ flight env activate modules
<modules> [flight@gateway1 ~]$
```

- Check that modules can be run:

```
<modules> [flight@gateway1 ~]$ module --version
Modules Release 4.3.0 (2019-07-26)
```

2.13.2 Installing Software - General Overview

Unlike the other *package ecosystems*, modules provides the ecosystem management tool but not any package management tools. Therefore, with the modules ecosystem, you are free to compile and install software in a module compatible manner.

Module files can be installed to `~/.local/share/flight/env/modules+default/modulefiles` (for local modules ecosystems) or to `/opt/apps/flight/env/modules+global/modulefiles` (for global modules ecosystems).

Note: If the modules ecosystem has been installed with a *custom ecosystem name* then the path will not be `modules+default`/`modules+global` but instead `modules+mycustomname`

For more information on building software for modules, see the modulefile reference and build documentation for the chosen software.

2.13.3 Installing and Running Perl

Compile Perl from Source

- Download perl 5.30.1 source:

```
<modules> [flight@gateway1 ~]$ wget https://www.cpan.org/src/5.0/perl-5.30.1.tar.
↪gz
```

- Decompress the source files:

```
<modules> [flight@gateway1 ~]$ tar -xzf perl-5.30.1.tar.gz
```

- Configure the software to install to a localperl directory:

```
<modules> [flight@gateway1 ~]$ cd perl-5.30.1
<modules> [flight@gateway1 ~]$ ./Configure -des -Dprefix=$HOME/localperl
```

- Compile and install perl:

```
<modules> [flight@gateway1 ~]$ make
<modules> [flight@gateway1 ~]$ make install
```

Create Modulefile and Test

- Create the perl modulefile:

```
<modules> [flight@gateway1 ~]$ cat << EOF > ~/.local/share/flight/env/
↪modules+default/modulefiles/perl-5.30.1
#%Module1.0
proc ModulesHelp { } {
global dotversion

puts stderr "\tPerl 5.30.1"
}

module-whatis "Perl 5.30.1"
conflict perl
prepend-path PATH ~/localperl/bin
prepend-path LD_LIBRARY_PATH ~/localperl/lib
prepend-path LIBRARY_PATH ~/localperl/lib
prepend-path MANPATH ~/localperl/man
EOF
```

- Check install location:

```
<modules> [flight@gateway1 ~]$ module load perl-5.30.1
<modules> [flight@gateway1 ~]$ which perl
~/localperl/bin/perl
```

- Install perl library (this may prompt for initial `cpan` configuration, once configuration is complete then the library will be installed):

```
<modules> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal logger. Log::Log4perl recommended for better logging
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Wed, 11 Mar 2020 15:29:03 GMT
<-- snip -->
Appending installation info to /home/flight/localperl/lib/5.30.1/x86_64-linux/
↪perllocal.pod
  CAPOEIRAB/File-Slurp-9999.30.tar.gz
  /usr/bin/make install  -- OK
```

- Check installation worked:

```
<modules> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal logger. Log::Log4perl recommended for better logging
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Wed, 11 Mar 2020 15:29:03 GMT
File::Slurp is up to date (9999.30).
```

2.14 Singularity: Usage Example

2.14.1 Creating and Using Ecosystem

Flight Env provides quick setup methods to create a singularity software ecosystem.

To install and use singularity:

Note: If installing singularity for a user then there are a number of restrictions and additional steps to consider in configuring the environment. See the `Personal Environment` section of `flight env info singularity`.

- *Activate the flight system*
- Create the singularity installation for the user:

```
[flight@gateway1 ~]$ flight env create singularity
Creating environment singularity@default
   >  Verifying prerequisites
   >  Fetching prerequisite (squashfs)
   >  Extracting prerequisite (squashfs)
   >  Building prerequisite (squashfs)
   >  Installing prerequisite (squashfs)
   >  Fetching prerequisite (go)
   >  Extracting prerequisite (go)
   >  Fetching prerequisite (singularity)
   >  Extracting prerequisite (singularity)
   >  Building prerequisite (singularity)
   >  Installing prerequisite (singularity)
   >  Creating environment (singularity@default)
Environment singularity@default has been created
```

- Activate the singularity ecosystem:

```
[flight@gateway1 ~]$ flight env activate singularity
<singularity> [flight@gateway1 ~]$
```

- Check that singularity can be run:

```
<singularity> [flight@gateway1 ~]$ singularity --version
singularity version 3.2.1
```

2.14.2 Installing and Running Perl

An example workflow using perl is demonstrated below.

Note: The perl container is built from a docker container which can be searched for in the docker hub. To search the singularity container library, use `singularity search SEARCHTERM`.

- Install specific version:

```
<singularity> [flight@gateway1 ~]$ singularity build --sandbox perl_5.30.simg␣
↪docker://perl:5.30
INFO:    Starting build...
Getting image source signatures
Copying blob␣
↪sha256:4ae16bd4778367b46064f39554128dd2fda2803a5747fddeff74059f353391c9
 48.05 MiB / 48.05 MiB [=====================================================] 0s
Copying blob␣
↪sha256:bbab4ec87ac4f89eaabdf68dddbd1dd930e3ad43bded38d761b89abf9389a893
 7.44 MiB / 7.44 MiB [=======================================================] 0s
<-- snip -->
Writing manifest to image destination
Storing signatures
INFO:    Creating sandbox directory...
INFO:    Build complete: perl_5.30.simg
```

- Check installation location:

```
<singularity> [flight@gateway1 ~]$ singularity exec perl_5.30.simg which perl
/usr/local/bin/perl
```

- Install perl library (this may prompt for initial `cpan` configuration, once configuration is complete then the library will be installed):

```
<singularity> [flight@gateway1 ~]$ singularity exec -w cpan File::Slurp
INFO:    Convert SIF file to sandbox...
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
    LANGUAGE = (unset),
    LC_ALL = (unset),
    LC_CTYPE = "en_GB.UTF-8",
    LANG = "en_GB.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Loading internal null logger. Install Log::Log4perl for logging messages
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Wed, 11 Sep 2019 13:29:02 GMT
Running install for module 'File::Slurp'
<-- snip -->
```

- Check installation worked:

```
<singularity> [flight@gateway1 ~]$ singularity exec perl_5.30.simg cpan␣
↪File::Slurp
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
    LANGUAGE = (unset),
    LC_ALL = (unset),
    LC_CTYPE = "en_GB.UTF-8",
    LANG = "en_GB.UTF-8"
    are supported and installed on your system.
```

(continues on next page)

(continued from previous page)

```
perl: warning: Falling back to the standard locale ("C").
Loading internal logger. Log::Log4perl recommended for better logging
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Wed, 11 Sep 2019 13:29:02 GMT
File::Slurp is up to date (9999.27).
```

2.15 Spack: Usage Example

2.15.1 Creating and Using Ecosystem

Flight Env provides quick setup methods to create a spack software ecosystem.

To install and use spack:

- *Activate the flight system*
- Create the spack installation for the user:

```
[flight@gateway1 ~]$ flight env create spack
Creating environment spack@default
   >  Verifying prerequisites
   >  Fetching prerequisite (spack)
   >  Extracting Spack hierarchy (spack@default)
   >  Bootstrapping Spack environment (spack@default)
Environment spack@default has been created
```

- Activate the spack ecosystem:

```
[flight@gateway1 ~]$ flight env activate spack
<spack> [flight@gateway1 ~]$
```

- Check that spack can be run:

```
<spack> [flight@gateway1 ~]$ spack --version
spack 0.12.1
```

2.15.2 Installing and Running Perl

An example workflow using perl is demonstrated below.

- View available versions:

```
<spack> [flight@gateway1 ~]$ spack list perl
==> 148 packages.
perl                          perl-extutils-pkgconfig     perl-math-cdf
↪         perl-sub-uplevel
perl-algorithm-diff           perl-file-copy-recursive    perl-math-cephes
↪         perl-svg
perl-app-cmd                  perl-file-listing           perl-math-matrixreal
↪         perl-swissknife
perl-array-utils              perl-file-pushd             perl-module-build
↪         perl-task-weaken
perl-b-hooks-endofscope       perl-file-sharedir-install  perl-module-
↪implementation       perl-term-readkey
```

(continues on next page)

(continued from previous page)

```
<-- snip -->

<spack> [flight@gateway1 ~]$ spack info perl
Package:   perl

Description:
    Perl 5 is a highly capable, feature-rich programming language with over
    27 years of development.

Homepage: http://www.perl.org

Tags:
    None

Preferred version:
    5.26.2     http://www.cpan.org/src/5.0/perl-5.26.2.tar.gz

Safe versions:
    5.28.0     http://www.cpan.org/src/5.0/perl-5.28.0.tar.gz
    5.26.2     http://www.cpan.org/src/5.0/perl-5.26.2.tar.gz
```

- Install specific version:

```
<spack> [flight@gateway1 ~]$ spack install perl@5.26.2
==> Installing pkgconf
==> Searching for binary cache of pkgconf
==> Warning: No Spack mirrors are currently configured
==> No binary for pkgconf found: installing from source
==> Fetching http://distfiles.alpinelinux.org/distfiles/pkgconf-1.4.2.tar.xz
<-- snip -->
==> Successfully installed perl
  Fetch: 0.83s.  Build: 2m 31.21s.  Total: 2m 32.04s.
[+] /home/flight/.local/share/flight/env/spack+default/opt/spack/linux-centos7-
↪x86_64/gcc-4.8.5/perl-5.26.2-wavwojlef7lshvx2awf4zze2lrx5l7l4
```

- Check installation location:

```
<spack> [flight@gateway1 ~]$ module load perl-5.26.2-gcc-4.8.5-wavwojl
<spack> [flight@gateway1 ~]$ which perl
~/.local/share/flight/env/spack+default/opt/spack/linux-centos7-x86_64/gcc-4.8.5/
↪perl-5.26.2-wavwojlef7lshvx2awf4zze2lrx5l7l4/bin/perl
```

- Install perl library (this may prompt for initial `cpan` configuration, once configuration is complete then the library will be installed):

```
<spack> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal null logger. Install Log::Log4perl for logging messages
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Wed, 11 Sep 2019 14:41:02 GMT
Running install for module 'File::Slurp'
<-- snip -->
```

- Check installation worked:

```
<spack> [flight@gateway1 ~]$ cpan File::Slurp
Loading internal null logger. Install Log::Log4perl for logging messages
```

(continues on next page)

(continued from previous page)

```
Reading '/home/flight/.cpan/Metadata'
  Database was generated on Wed, 11 Sep 2019 14:41:02 GMT
File::Slurp is up to date (9999.27).
```