flight-appliance-docs Documentation

Release 1.0

Alces Flight Ltd

Jul 26, 2017

This site holds documentation designed to help users create a simple research compute environment using popular public and private cloud platforms. As well as launching and accessing the environment, guides and tutorials are included to help end-users install software applications, manage their data and run different workloads.

CHAPTER 1

License

CHAPTER 2

Prerequisites

We recommend that users wishing to use Flight Appliances have basic Linux skills. The ability to move about in a filesystem, copy and delete files, read and edit files on the command-line will be needed in order to get the best out of the Flight software.

Getting started as quickly as possible

This documentation is already designed to provide a quick-start guide, but if you've used clusters before then you might just need a *super-quick* guide to getting started with your Alces Flight Compute cluster. Here are just the basics, along with things you should probably look at in more detail when you have time. Useful commands are included with each step - please refer to the full documentation for further details.

1. Launch your personal Alces Flight Compute cluster on AWS or a compatible OpenStack private cloud. You will be asked various questions about how you want your cluster to look - most options are explained, with sensible defaults included.
2. Login to your cluster via SSH using your chosen username, and the SSH key registered with your cloud provider. Your cloud service should report the access IP address to login to once your cluster is launched. If you enabled cluster auto-scaling, compute nodes are automatically added/removed based on the status of your job-scheduler queue. Use `alces configure autoscaling disable` to turn off scaling if you'll be running jobs manually.
3. Start a graphical desktop session with the command `alces session start gnome`. Connect using a VNC client, with the password provided when you started the session. Multiple users can connect to the same session for collaborative projects. Use the `xrandr` command to change the screen resolution of a running session.
4. Your user has full `sudo` access, and `yum` is configured to install Linux packages. Use the `nodeattr -n nodes` command to see your list of compute node hostnames, and `pdsh -g nodes <command>` to run commands across all nodes in your cluster.
5. Your cluster has an NFS shared filesystem mounted across all nodes, which your home-directory is part of. Copy data to and from the cluster using SCP/SFTP. The `alces storage` commands provide access to object storage services including S3 and Dropbox.

6. Use the `alces gridware` command to view and install software. The `main` repository lists stable packages, with a larger list of applications available as part of the `volatile` repository.
7. Your cluster comes with an installation of the SGE job-scheduler; use the `alces template` command for a list of template job-scripts to get you started.
8. Terminate the cluster stack via your cloud provider when you've finished working. Remember to copy any data you want to keep off the cluster to secure storage before terminating it.

What is Alces Flight Compute?

Alces Flight Compute is a software appliance designed to help researchers and scientists build their own high-performance compute cluster quickly and easily. The basic structure provided for users is as follows:

- One login 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 a library of software applications

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

Your cluster 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 Flight clusters, 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 learn to work effectively in this way.

Who is it for?

Flight 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.

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 cluster, you need to remember good security practice, and look after both your personal and research data. Flight provides you with a personal, single-user cluster - please use responsibility. Flight is not indended as a replacement for your national super-computer centre.

If you're running Flight 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 cluster is any less secure than a cluster running in your basement. Start at the AWS Security Pages, and talk to a security expert if you're still unsure.

For Flight clusters running on OpenStack, you need to contact the administration team who gave you your OpenStack account. They are usually helpful and friendly people, so try and explain what you want to do as clearly and concisely as possible. If you are having problems accessing the cluster once it's launched, remember that you can always do a few trials runs on AWS first, just to make sure you've got the hang of it - the Flight software is the same, whatever platform it's running on.

How much does it cost?

Alces Flight Compute is an open-source software appliance and is freely available to users, released under the General Public License (GPL). Please see the Alces Flight EULA for details.

You are likely to incur infrastructure costs from your platform provider (i.e. AWS, or your local OpenStack team) if you are using their compute resources. This documentation will highlight the configuration points that can effect how much you might be charged, but a full cost estimate for your work is outside the scope of Flight documentation.

Your platform providers are there to help though - if in doubt, start small (e.g. a handful of nodes for a few hours) and review the costs before scaling your workload up. It's worth spending some time reviewing your costs over the longer term as well, as you may be able to reduce the bill if you run jobs at a different time of day, on different types of resources, or even in a different geographic region if your datasets allow.

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 cluster 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.

Where can I get help?

This documentation is designed to walk users through the first stages of creating their clusters, 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 clusters 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 Flight software - available online here. This website is designed to help users share their experiences of running Flight clusters, 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 Flight community support site is a great resource for helping with HPC cluster usage, but for software application support you're going to need to contact the developers of the packages themselves. Each software package installed by Flight comes with a link to the online home of the package (e.g. `module display apps/gromacs`), where you can highlight any issues to the package maintainers. 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.

If you're a big company or research group and want to pay for support delivered direct-to-you, then please contact us at info@alces-flight.com . We provide consultancy and targeted support services directly and via a network of partners - it's this that funds the open-source Flight projects.

Launching on AWS

Alces Flight Compute can be launched on the Amazon Web Services (AWS) public cloud platform to give you instant access to your own, private HPC cluster from anywhere in the world. You can choose what resources your cluster will start with (e.g. number of nodes, amount of memory, etc.), and for how long the cluster will run.

Prerequisites

There are some things that you need to get ready before you can launch your own cluster on AWS. They are:

- **Check client prerequisites** to make sure you have the software you need - see *What is Alces Flight Compute?*
- **Get yourself an AWS account**; this might be your personal account, or you may have a sub-account as part of your institution or company
- **Create an SSH keypair** for yourself in the region you want to run in. Follow this guide if you've not done this before.

Your AWS account must have appropriate permissions to do the following:

- Launch instances from Cloud Formation templates
- Create a VPC (virtual private cloud)
- Create subnets and allocate IP addresses
- Create an IAM permission

More details on AWS Identity and Access Management (IAM) are available here.

Creating your Cluster

How much will it cost?

The Alces Flight software appliance itself is free; however, you're likely to incur costs when running a cluster on AWS resources. Charges typically fall into the following categories:

- EC2 charges for running instances (your login and compute nodes)
- EBS charges for shared cluster filesystem capacity
- S3 charges for storing data as objects
- Data-egress charges for network traffic out of AWS
- Miscellaneous other charges (e.g. IP address allocation, DNS entry updates, etc.)

Most charges are made per unit (e.g. per compute node instance, or per GB of storage space) and per hour, often with price breaks for using more of a particular resource at once. A full breakdown of pricing is beyond the scope of this document, but there are several tools designed to help you estimate the expected charges; e.g.

- AWS Simple Monthly Calculator
- AWS TCO Calculator

Finding Alces Flight Compute on AWS

Sign-in to your AWS account, and navigate to the AWS Marketplace. Search for **Alces Flight** in the search box provided to find the Flight Compute product. Click on the *Continue* button to view details on how to launch.

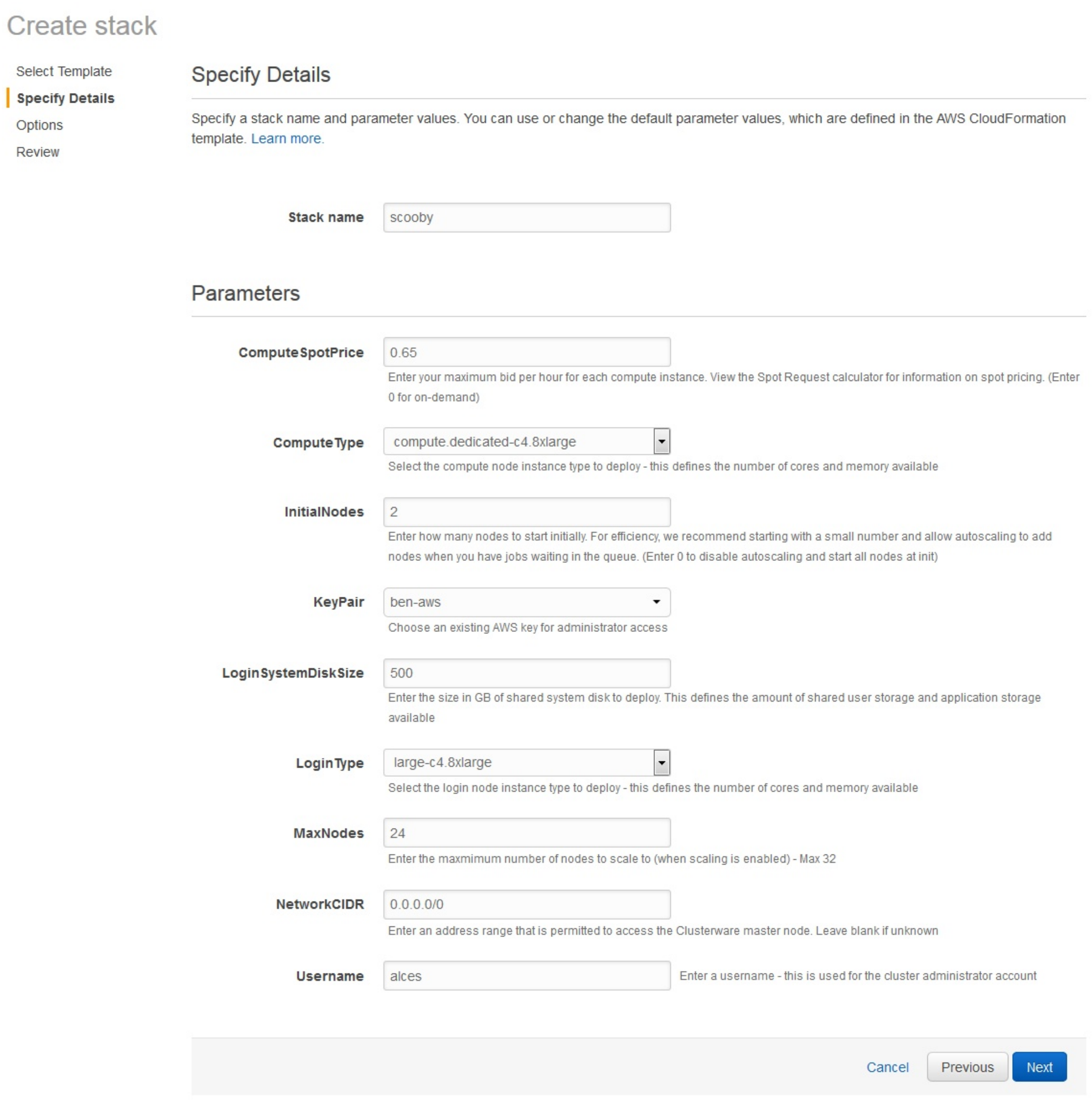
As well as an Amazon Machine Image (AMI), Flight Compute subscribers are provided with a Cloud-Formation template (CFT) that can be used to launch your own cluster rapidly after answering a few setup questions. Advanced users can also use the AMI directly with their own CFTs to provide more customised environments for specialised requirements. This documentation is designed to assist new users when launching with the CFT provided on the AWS Marketplace page.

How to answer Cloud-Formation questions

When you choose to start a Flight Compute cluster from AWS Marketplace, you will be prompted to answer a number of questions about what you want the environment to look like. Flight will automatically launch your desired configuration based on the answers you give. The questions you'll be asked are the following:

- **Stack name**; this is the name that you want to call your cluster. It's fine to enter **"cluster"** here if this is your first time, but entering something descriptive will help you keep track of multiple clusters if you launch more. Naming your cluster after colours (red, blue, orange), your favourite singer (clapton, toriamos, bieber) or Greek legends (apollo, thor, aphrodite) keep things more interesting. Avoid using spaces and punctuation, or names longer than 16 characters.
- **ComputeSpotPrice**; in this box, enter the maximum amount you agree to pay per compute node instance, in US dollars. Entering **0** (zero) in this box will cause Flight to use **on-demand** instances for compute nodes. See the section below on *On-demand and SPOT* instances for more details.
- **ComputeType**; use the drop-down box to choose what type of compute nodes you want to launch. All compute nodes will launch as the same type. Different types of nodes cost different amounts to run, and have different amounts of CPU-cores and memory - see the AWS Pricing Guide for more information. Node instances are grouped in the following ways:
  - **Type (compute/balanced/memory/gpu):**
    - Compute instances have 2GB of memory per core, and provide the fastest CPUs
    - Balanced instances have 4GB of memory per core, and are good all-round performers
    - Memory instances have 8GB of memory per core, and are useful for high-memory jobs
    - GPU instances have Nvidia CUDA GPU devices installed
  - **Size (small/medium/large/dedicated):**
    - Small, medium and large instances have 2, 4 or 8 CPU cores, and a fraction of a 10Gb Ethernet network link
    - Dedicated instances have access to a dedicated 10Gb Ethernet network link
- **FlightCustomBucket**; enter an S3 bucket containing customisation information for your cluster. Leave this option blank if you have no existing customisation data, or you are starting a new cluster.
- **FlightCustomProfiles**; enter the names of the customisation profiles to use, separated by spaces. Leave this option blank if you have no existing customisation data, or you are starting a new cluster.
- **InitialNodes**; enter the number of nodes you want to start immediately in this box. Entering any number here will enable auto-scaling of the cluster - Flight Compute will add more nodes when jobs are queued, and shutdown idle nodes when they have no jobs to process. Entering 0 (zero) in this box will disable auto-scaling, and start all cluster nodes immediately.

- **Keypair**; choose an existing AWS keypair to launch your Flight cluster with. If there are no keypairs in the list, check that you've already generated a keypair in the region you're launching in. You must have the private key available for the chosen keypair in order to login to your cluster.
- **LoginSystemDiskSize**; choose the size of your login node disk, which acts as the shared filesystem for your cluster. Requesting a larger size will give you more space for your data, but will cost more to run.
- **LoginType**; use the drop-down box to choose the AWS instance type for your login node. Larger sizes will perform better, while smaller sizes will be less expensive to run. Your login node is always created as an on-demand instance.
- **MaxNodes**; enter the maximum size that your cluster will scale to, up to a maximum of 32 nodes.
- **NetworkCIDR**; enter a network range that is permitted to access your cluster. This will usually be the IP address of your system on the Internet; ask your system administrator for this value, or use a web search to find out. If you want to be able to access your cluster from anywhere on the Internet, enter "0.0.0.0/0" in this box.
- **Username**; enter the username you want to use to connect to the cluster. Flight will automatic create this user on the cluster, and add your public SSH key to the user.



Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. Learn more.

Stack name: scooby

Parameters

ComputeSpotPrice: 0.65
Enter your maximum bid per hour for each compute instance. View the Spot Request calculator for information on spot pricing. (Enter 0 for on-demand)

ComputeType: compute.dedicated-c4.8xlarge
Select the compute node instance type to deploy - this defines the number of cores and memory available

InitialNodes: 2
Enter how many nodes to start initially. For efficiency, we recommend starting with a small number and allow autoscaling to add nodes when you have jobs waiting in the queue. (Enter 0 to disable autoscaling and start all nodes at init)

KeyPair: ben-aws
Choose an existing AWS key for administrator access

LoginSystemDiskSize: 500
Enter the size in GB of shared system disk to deploy. This defines the amount of shared user storage and application storage available

LoginType: large-c4.8xlarge
Select the login node instance type to deploy - this defines the number of cores and memory available

MaxNodes: 24
Enter the maxmimum number of nodes to scale to (when scaling is enabled) - Max 32

NetworkCIDR: 0.0.0.0/0
Enter an address range that is permitted to access the Clusterware master node. Leave blank if unknown

Username: alces
Enter a username - this is used for the cluster administrator account

When all the questions are answered, click the **Next** button to proceed. Enter any tags you wish to use to identify instances in your environment on the next page, then click the **Next** button again. On the review page, read through the answers you've provided and correct any mistakes - click on the *Capabilities* check-box to authorize creations of an IAM role to report cluster performance back to AWS, and click on the **Create** button.

Your personal compute cluster will then be created. While on-demand instances typically start within in few minutes, SPOT based instances may take longer to start, or may be queued if the SPOT price you entered is less than the current price.

On-demand vs SPOT instances

The AWS EC2 service supports a number of different charging models for launching instances. The quick-start Cloud-formation template included with Alces Flight Compute in AWS Marketplace allows users to choose between two different models:

- On-demand instances; instances are launched immediately at a fixed hourly price. Once launched, your instance will not normally be terminated unless you choose to stop it.
- SPOT instances; instances are requested with a bid-price entered by the end-user which represents the maximum amount they want to pay for them per hour. If public demand for this instance type allows, instances will be launched at the current SPOT price, which is typically much lower than the equivalent on-demand price. As demand increases for the instance type increases, so the cost per hour charged to users also increases. AWS will automatically stop any instances (or delay starting new ones) if the current SPOT price is higher than the maximum amount users want to pay for them.

SPOT instances are a good way to pay a lower cost for cloud computing for non-urgent workloads. If SPOT compute node instances are terminated in your cluster, any running jobs will be lost - the nodes will also be automatically removed from the queue system to ensure no new jobs attempt to start on them. Once the SPOT price becomes low enough for your instances to start again, your compute nodes will automatically restart and rejoin the cluster.

The Cloud-formation templates provided for Alces Flight Compute via AWS Marketplace will not launch a login node instance on the SPOT market - **login nodes are always launched as on-demand instances**, and are immune from fluctuating costs in the SPOT market.

Using an auto-scaling cluster

An auto-scaling cluster automatically reports the status of the job scheduler queue to AWS to allow idle compute nodes to be shut-down, and new nodes to be started when jobs are queuing. Auto-scaling is a good way to manage the size of your ephemeral cluster automatically, and is useful if you want to run a number of unattended jobs, and minimise costs after the jobs have finished by terminating unused resources.

Your Alces Flight compute cluster will never scale larger than the maximum number of instances entered at launch time. The cluster will automatically scale down to a single compute node when idle, or be reduced to zero nodes if you are using SPOT based compute nodes, and the price climbs higher than your configured maximum.

If you are running jobs manually (i.e. not through the job-scheduler), you may wish to disable autoscaling to prevent nodes not running scheduled jobs from being shutdown. This can be done by entering `0` (zero) in the **ComputeSpotPrice** when launching your Flight Compute cluster via AWS Marketplace, or using the command `alces configure autoscaling disable` command when logged in to the cluster login node.

Accessing your cluster

Once your cluster has been launched, the login node will be accessible via SSH from the IP address range you entered in the **NetworkCIDR**. If you entered `0.0.0.0/0` as the **NetworkCIDR**, your login node will be accessible from any IP address on the Internet. Your login node's public IP address is reported by the AWS Cloud-formation template, along with the username you must use to login with your keypair.

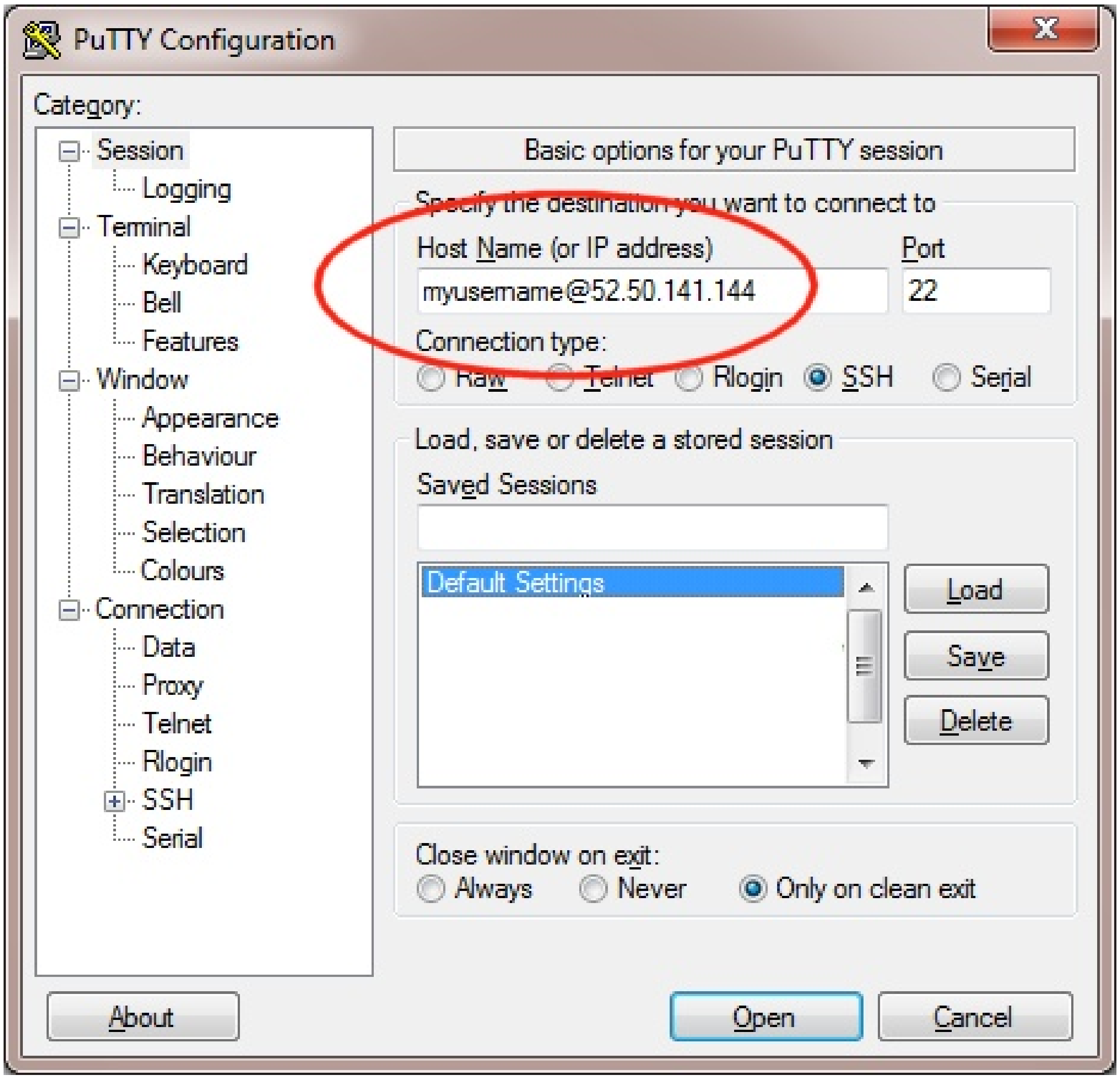
To access the cluster login node from a Linux or Mac client, use the following command:

- `ssh -i mypublickey.pub myusername@52.50.141.144`

Where:

- `mypublickey.pub` is the name of your public SSH key you selected when launching the cluster
- `myusername` is the username you entered when launching the cluster
- `52.50.141.144` is the Access-IP address reported by the AWS console after your cluster has been launched

If you are accessing from a Windows client using the Putty utility, enter the username and IP address of the cluster login node in the "Host Name" box provided:



Specify the destination you want to connect to

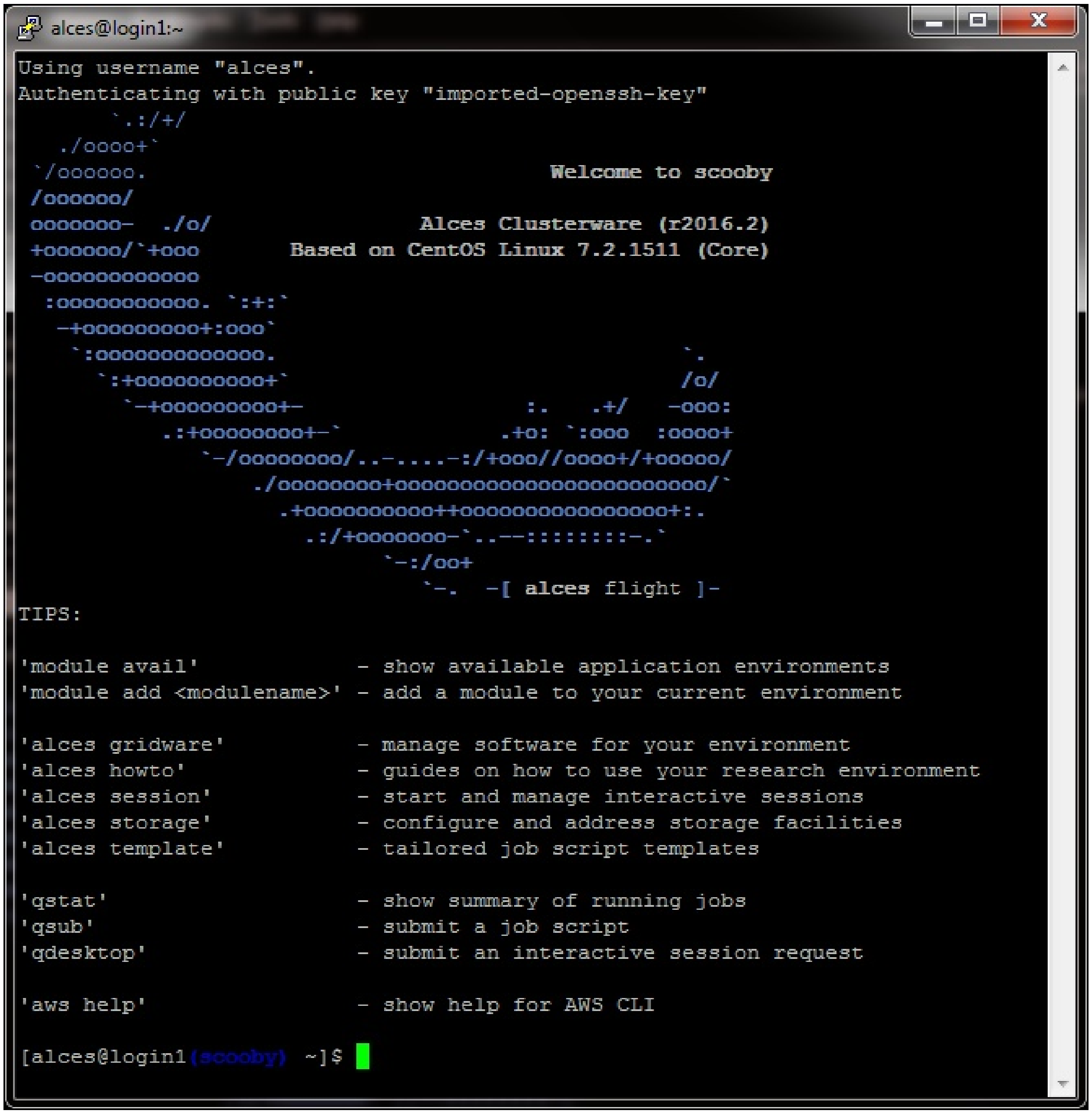
Host Name (or IP address): myusername@52.50.141.144

Port: 22

Connection type: Raw / Telnet / Rlogin / [✓ SSH] / Serial

Close window on exit: Always / Never / [✓ Only on clean exit]

The first time you connect to your cluster, you will be prompted to accept a new server SSH hostkey. This happens because you've never logged in to your cluster before - it should only happen the first time you login; click **OK** to accept the warning. Once connected to the cluster, you should be logged in to the cluster login node as your user.



```
Using username "alces".
Authenticating with public key "imported-openssh-key"
        `.:/+/
    ./oooo+`
  `/oooooo.                              Welcome to scooby
  /oooooo/
  ooooooo-   ./o/                   Alces Clusterware (r2016.2)
  +oooooo/`+ooo         Based on CentOS Linux 7.2.1511 (Core)
  -oooooooooooo
   :ooooooooooo. `:+:`
    -+ooooooooo+:ooo`
     `:ooooooooooooo.                                   `.
       `:+oooooooooo+`                                  /o/
         `-+oooooooooo+-                   :.    .+/    -ooo:
            .:+oooooooo+-`              .+o:  `:ooo   :oooo+
               `-/ooooooo/..-....-:/+ooo//oooo+/+ooooo/
                   ./ooooooo+oooooooooooooooooooooooo/`
                     .+oooooooooo++oooooooooooooooo+:.
                       .:/+ooooooo-`..--::::::::-.`
                             `-:/oo+
                                `-.  -[ alces flight ]-
TIPS:

'module avail'             - show available application environments
'module add <modulename>'  - add a module to your current environment

'alces gridware'           - manage software for your environment
'alces howto'              - guides on how to use your research environment
'alces session'            - start and manage interactive sessions
'alces storage'            - configure and address storage facilities
'alces template'           - tailored job script templates

'qstat'                    - show summary of running jobs
'qsub'                     - submit a job script
'qdesktop'                 - submit an interactive session request

'aws help'                 - show help for AWS CLI

[alces@login1(scooby) ~]$
```

Terminating the cluster

Your cluster login node will continue running until you terminate it via the AWS web console. If you are running an auto-scaling cluster, compute nodes will automatically be added and taken away up to the limits you specified depending on the number of jobs running and queued in the job-scheduler. When you have finished running your workloads, navigate to the Cloud-formation console, select the name of your cluster from the list of running stacks, and click **Delete stack** from the actions menu.

Over the next few minutes, your cluster login and compute nodes will be terminated. Any data held on EBS will be erased, with storage volumes being wiped and returned to the AWS pool. **Ensure that you have downloaded data that you want to keep to your client machine, or stored in safely in an object storage service before terminating your cluster.**

See - *Working with data and files* for more information on storing your data.

Launching on OpenStack

Alces Flight Compute can be launched on your local OpenStack private cloud platform to give you access to your own, private HPC cluster using your on-premise infrastructure.

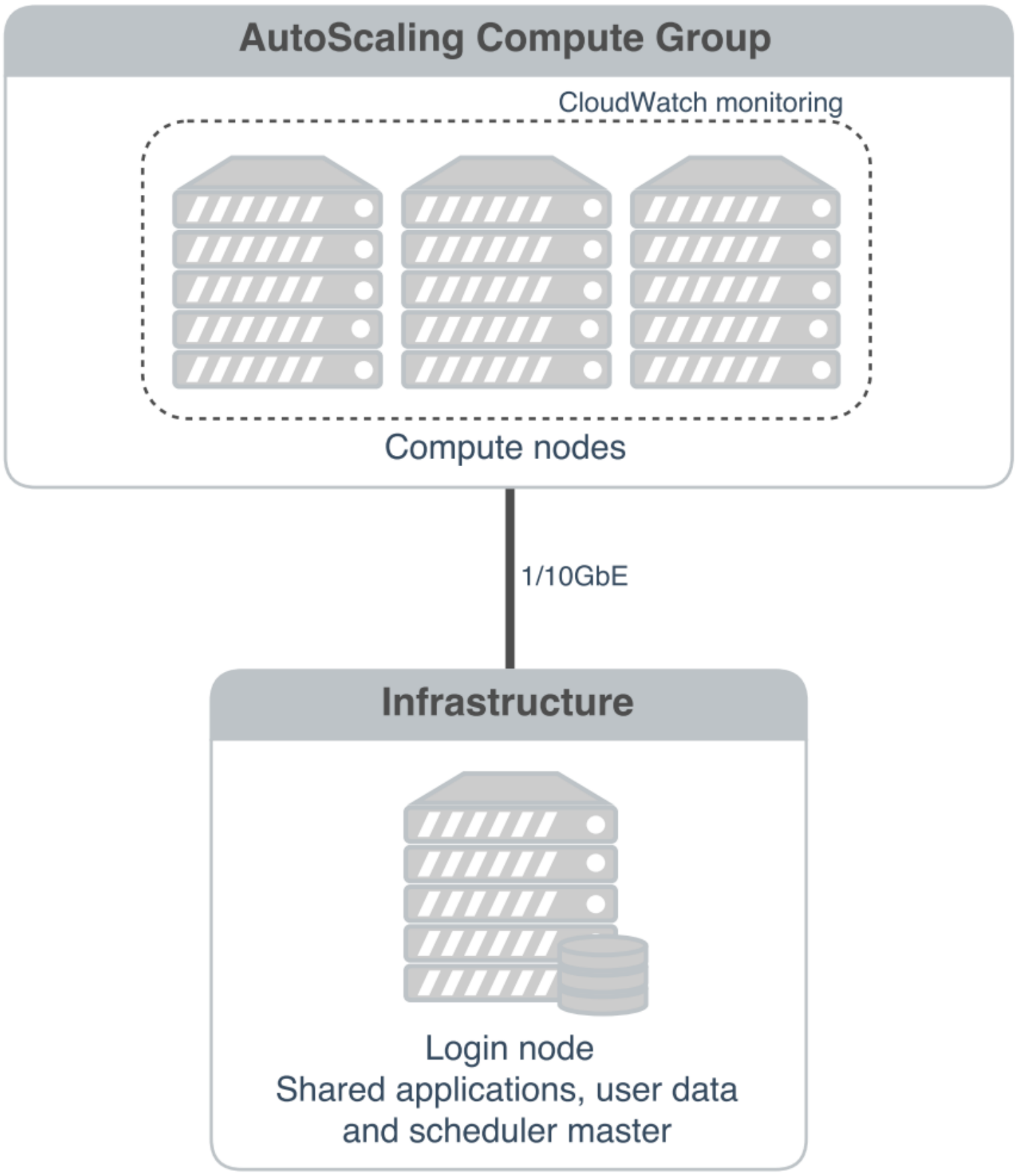
More instructions will appear here as this functionality is tested and made available to end-users.

Basic cluster operation

Logging in

You can access the login node for your private Flight Compute cluster using SSH from the network ranges you permitted at launch time. You will need to use the SSH keypair configured for the cluster in order to access it.

In cluster launched using the Flight Compute AWS Marketplace Cloud-formation template, compute nodes do not have Internet-addressable IP addresses. In order to access individual compute nodes, you must first login to the cluster login node.



When you login to the cluster via SSH, you are automatically placed in your home-directory. This area is shared across all compute nodes in the cluster, and is mounted in the same place on every compute. Data copied to the cluster or created in your home-directory on the login node is also accessible from all compute nodes.

Becoming the root user

Most cluster operations, including starting applications and running jobs, should be performed as the user created when the Flight Compute cluster was launched from AWS Marketplace. However - for some privileged operations, users may need to change to being the root user. Users can prefix any command they want to run as the root user with the `sudo` command; e.g.

```
sudo yum install screen
```

For security reasons, SSH login as the root user is not permitted to a Flight Compute environment. To get a Linux shell with root privileges, please login as your standard user then execute the command `sudo -s`.

Users must exercise caution when running commands as root, as they have the potential to disrupt cluster operations.

Finding the names of your compute nodes

An Alces Flight Compute cluster may contain any number of compute nodes which may be automatically started and stopped in response to the workloads being processed. The hostnames of compute nodes are set automatically at launch time - your set of compute nodes may change during the life span of your cluster login node. Flight Compute automatically updates a list of compute node names in response to the changing size of your cluster, and uses them to populate a *genders* group called **nodes**.

Users can find the names of their compute nodes by using the `nodeattr` command; e.g.

- `nodeattr -s nodes` - shows a space-separated list of current compute node hostnames
- `nodeattr -c nodes` - shows a comma-separated list of current compute node hostnames
- `nodeattr -n nodes` - shows a new-line-separated list of current compute node hostnames

The login node hostname for Flight Compute clusters launched from AWS Marketplace is always `login1`.

Moving between login and compute nodes

Flight Compute clusters automatically configure a trust relationship between login and compute nodes in the same cluster to allow users to login between nodes via SSH without a password. This configuration allows moving quickly and easily between nodes, and simplifies running large-scale jobs that involve multiple nodes. From the command line, a user can simply use the `ssh <node-name>` command to login to one of the compute nodes from the login node. For example, to login to a compute node named `host-22-33-11-123` from the login node, use the command:

```
ssh host-22-33-11-123
```

Use the `logout` command (or press **CTRL+D**) to exit the compute node and return to the login node.

Using PDSH

Users can run a command across all compute nodes at once using the `pdsh` command. This can be useful if users want to make a change to all nodes in the cluster - for example, installing a new software package. The `pdsh` command can take a number of parameters that control how commands are processed; for example:

- **`pdsh -g cluster uptime`**
  - executes the `uptime` command on all available compute and login nodes in the cluster
- **`pdsh -g nodes 'sudo yum -y install screen'`**
  - use `yum` to install the `screen` package as the root user on all compute nodes
- **`pdsh -g nodes -f 1 df -h /tmp`**
  - executes the command `df -h /tmp` on all compute nodes of the cluster, one at a time (fanout=1)
- **`pdsh -w ip-10-75-0-235,ip-10-75-0-66 which ldconfig`**

    - runs the `which ldconfig` command on two named nodes only

Working with data and files

Organising data on your cluster

Shared filesystem

Your Flight Compute cluster includes a shared home filesystem which is mounted across the login and all compute nodes. Files copied to this area are available via the same absolute path on all cluster nodes. The size of this area is controlled by the setting used when you created your cluster. The shared filesystem is typically used for job-scripts, input and output data for the jobs you run.

Users must make sure that they copy data they want to keep off the shared filesystem before the Flight Compute cluster is terminated. This documentation provides example methods for copying data back to your local client system, or storing it in the **AWS Simple Storage Service (S3)**.

Your home directory

The shared filesystem includes the home-directory area for the user you created when your cluster was launched. Linux automatically places users in their home-directory when they login to a node. By default, Flight Compute will create your home-directory under the `/home/` directory, named after your username. For example, if your user is called **jane**, then your home-directory will have the absolute path `/home/jane/`.

The Linux command line will accept the ~ (*tilde*) symbol as a substitute for the currently logged-in users' home-directory. The environment variable `$HOME` is also set to this value by default. Hence, the following three commands are all equivalent when logged in as the user **jane**:

- `ls /home/jane`
- `ls ~`
- `ls $HOME`

The **root** user in Linux has special meaning as a privileged user, and does not have a shared home-directory across the cluster. The **root** account on all nodes has a home-directory in `/root`, which is separate for every node. For security reasons, users are not permitted to login to a node as the root user directly - please login as a standard user and use the `sudo` command to get privileged access.

Local scratch storage

Your compute nodes have an amount of disk space available to store temporary data under the `/tmp` mount-point. The size of this area will depend on the type and size of instance you selected at the time your cluster was launched. This area is intended for temporary data created during compute jobs, and shouldn't be used for long-term data storage. Compute nodes are configured to automatically clear up temporary space automatically, removing orphan data left behind by jobs. In addition, an auto-scaling cluster may automatically terminate idle nodes, resulting in the loss of any files stored in local scratch space.

Users must make sure that they copy data they want to keep back to the shared filesystem after compute jobs have been completed.

Copying data between nodes

Flight Compute cluster login and compute nodes all mount the shared filesystem, so it is not normally necessary to copy data directly between nodes in the cluster. Users simply need to place the data to be shared in their home-directory on the login node, and it will be available on all compute nodes in the same location.

If necessary, users can use the `scp` command to copy files from the compute nodes to the login node; for example:

- `scp ip-10-75-0-235:/tmp/myfile.txt .`

Alternatively, users could login to the compute node (e.g. `ssh ip-10-75-0-235`) and copy the data back to the shared filesystem on the node:

```
ssh ip-10-75-0-235
cp /tmp/myfile ~/myfile
```

Copying data files to the cluster

Many compute workloads involve processing data on the cluster - users often need to copy data files to the cluster for processing, and retrieve processed data and results afterwards. This documentation describes a number of methods of working with data on your cluster, depending on how users prefer to transfer it.

Using command-line tools to copy data

The cluster login node is accessible via SSH, allowing use of the `scp` and `sftp` commands to transfer data from your local client machine. Linux and Mac users can use in-built SSH support to copy files; e.g.

- **To copy file mydata.zip to your cluster on IP address 52.48.62.34, use the command:** `scp -i mykeyfile.pub mydata.zip jane@52.48.62.34:.`
  - replace `mykeyfile.pub` with the name of your SSH public key
  - replace `jane` with your username on the cluster

Windows users can download and install the pscp command to perform the same operation: `pscp -i mykeyfile.ppk mydata.zip jane@52.48.62.34:/home/jane/.`

Both the `scp` and the `pscp` commands take the parameter `-r` to recursively copy entire directories of files to the cluster.

To retrieve files from the cluster, simply specify the location of the remote file first in the `scp` command, followed by the location on the local system to put the file; e.g.

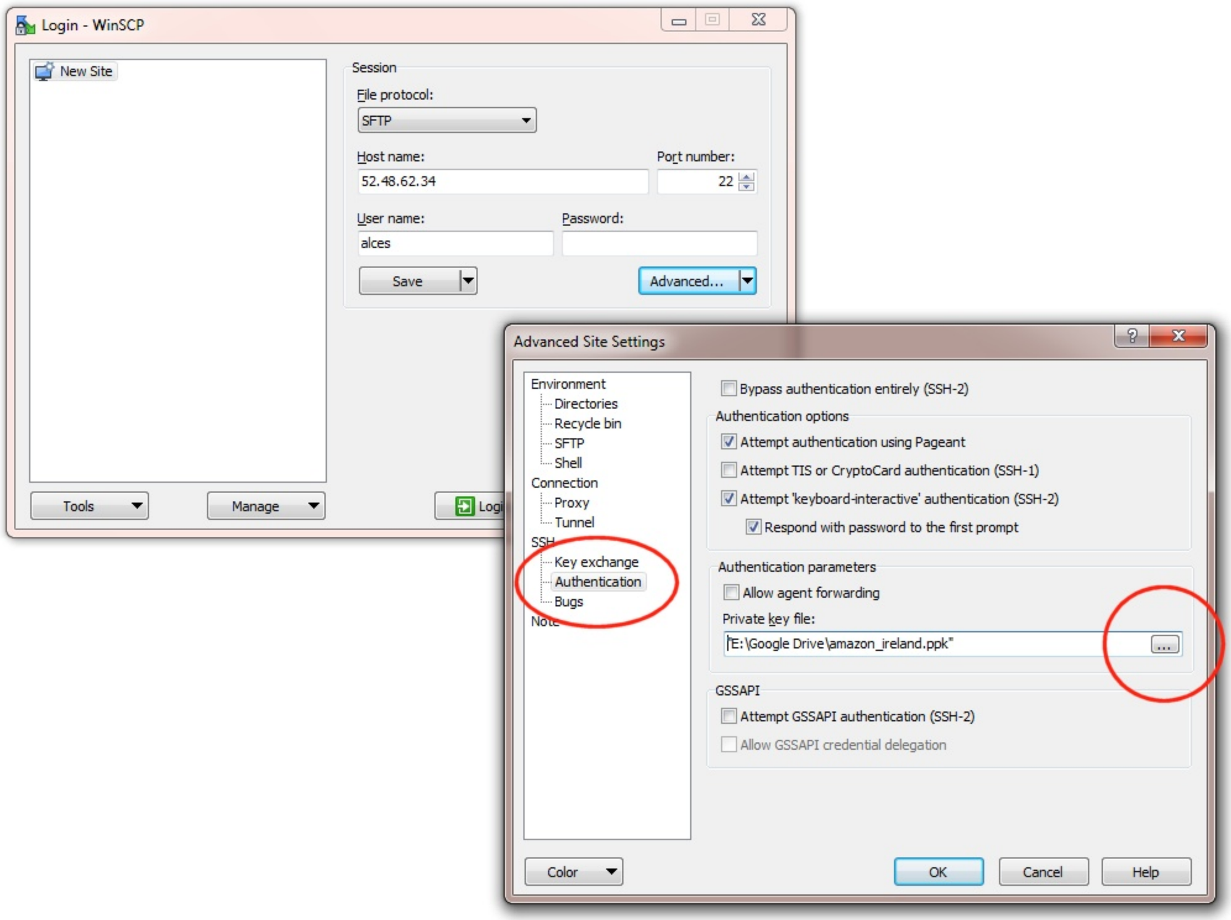
- **To copy file myresults.zip from your cluster on IP address 52.48.62.34 to your local Linux or Mac client:** `scp -i mykeyfile.pub jane@52.48.62.34:/home/jane/myresults.zip .`

Using a graphical client to copy data

There are also a number of graphical file-management interfaces available that support the SSH/SCP/SFTP protocols. A graphical interface can make it easier for new users to manage their data, as they provide a simple drag-and-drop interface that helps to visualise where data is being stored. The example below shows how to configure the WinSCP utility on a Windows client to allow data to be moved to and from a cluster.

- On a Windows client, download and install WinSCP
- Start WinSCP; in the **login** configuration box, enter the IP address of your Flight Compute cluster login node in the `Host name` box

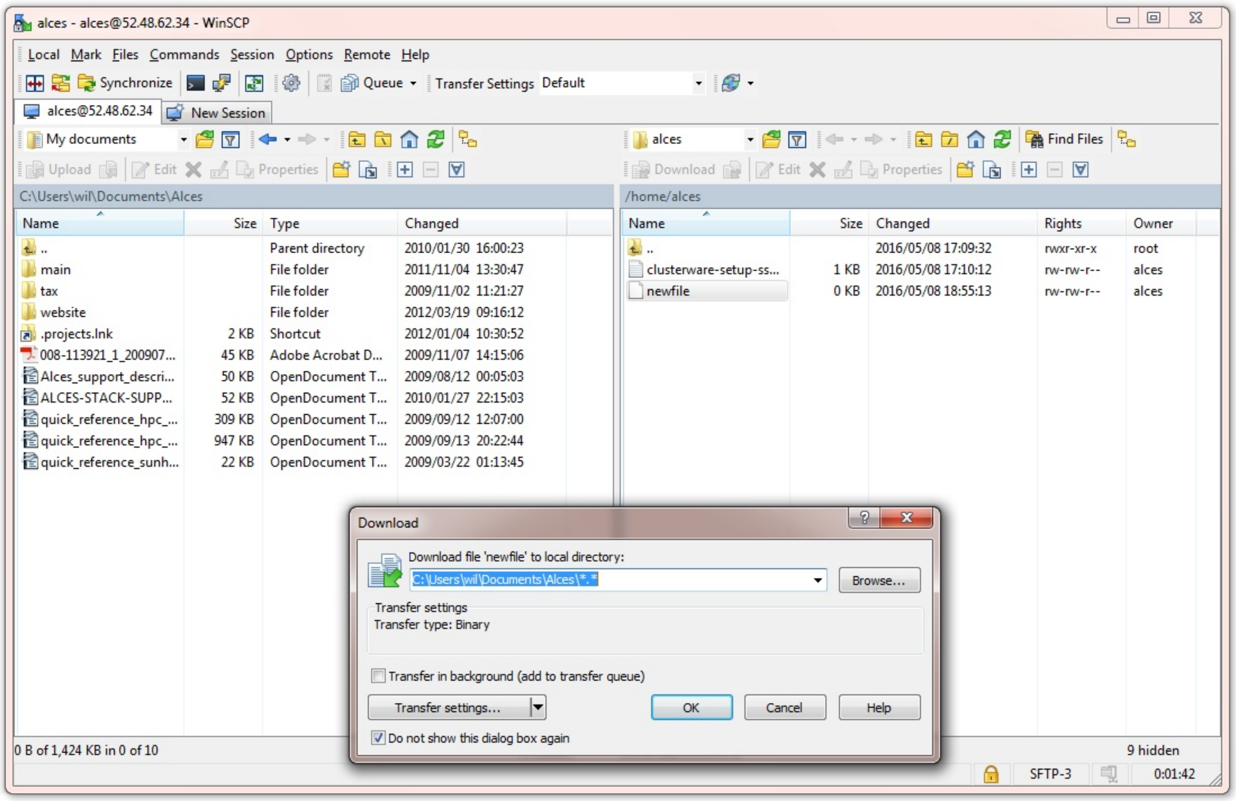
- Enter the username you configured for your cluster in the `User name` box
- Click on the `Advanced` box and navigate to the `SSH` sub-menu, and the `Authentication` item
- In the `Private key file` box, select your AWS private key, and click the `OK` box.



Session
File protocol: SFTP
Host name: 52.48.62.34
Port number: 22
User name: alces
Password:

[ ] Bypass authentication entirely (SSH-2)
Authentication options
[✓] Attempt authentication using Pageant
[ ] Attempt TIS or CryptoCard authentication (SSH-1)
[✓] Attempt 'keyboard-interactive' authentication (SSH-2)
[✓] Respond with password to the first prompt
Authentication parameters
[ ] Allow agent forwarding
Private key file: "E:\Google Drive\amazon_ireland.ppk"
GSSAPI
[ ] Attempt GSSAPI authentication (SSH-2)
[ ] Allow GSSAPI credential delegation

- Optionally click the `Save` button and give this session a name
- Click the `Login` button to connect to your cluster
- Accept the warning about adding a new server key to your cache; this message is displayed only once when you first connect to a new cluster
- WinSCP will login to your cluster; the window shows your local client machine on the left, and the cluster on the right
- To copy files to the cluster from your client, click and drag them from the left-hand window and drop them on the right-hand window
- To copy files from the cluster to your client, click and drag them from the right-hand window and drop them on the left-hand window



C:\Users\wil\Documents\Alces

Name	Size	Type	Changed
..		Parent directory	2010/01/30 16:00:23
main		File folder	2011/11/04 13:30:47
tax		File folder	2009/11/02 11:21:27
website		File folder	2012/03/19 09:16:12
.projects.lnk	2 KB	Shortcut	2012/01/04 10:30:52
008-113921_1_200907...	45 KB	Adobe Acrobat D...	2009/11/07 14:15:06
Alces_support_descri...	50 KB	OpenDocument T...	2009/08/12 00:05:03
ALCES-STACK-SUPP...	52 KB	OpenDocument T...	2010/01/27 22:15:03
quick_reference_hpc_...	309 KB	OpenDocument T...	2009/09/12 12:07:00
quick_reference_hpc_...	947 KB	OpenDocument T...	2009/09/13 20:22:44
quick_reference_sunh...	22 KB	OpenDocument T...	2009/03/22 01:13:45

/home/alces

Name	Size	Changed	Rights	Owner
..		2016/05/08 17:09:32	rwxr-xr-x	root
clusterware-setup-ss...	1 KB	2016/05/08 17:10:12	rw-rw-r--	alces
newfile	0 KB	2016/05/08 18:55:13	rw-rw-r--	alces

Download

Download file 'newfile' to local directory:
C:\Users\wil\Documents\Alces\*.*

Browse...

Transfer settings
Transfer type: Binary

☐ Transfer in background (add to transfer queue)

Transfer settings... | OK | Cancel | Help

☑ Do not show this dialog box again

0 B of 1,424 KB in 0 of 10 | 9 hidden | SFTP-3 | 0:01:42

The amount of time taken to copy data to and from your cluster will depend on a number of factors, including:

- The size of the data being copied
- The speed of your Internet link to the cluster; if you are copying large amounts of data, try to connect using using a wired connection rather than wireless
- The type and location of your cluster login node instance

Object storage for archiving data

As an alternative to copying data back to your client machine, users may prefer to upload their data to a cloud-based object storage service instead. Flight Compute clusters include tools for accessing data stored in the AWS S3 object storage service, as well as the Dropbox cloud storage service. Benefits of using an cloud-based storage service include:

- Data is kept safe and does not have to be independantly backed-up
- Storage is easily scalable, with the ability for data to grow to practically any size
- You only pay for what you use; you do not need to buy expansion room in advance
- Storage service providers often have multiple tiers available, helping to reduce the cost of storing data
- Data storage and retrieval times may be improved, as storage service providers typically have more bandwidth than individual sites
- Your company, institution or facility may receive some storage capacity for free which you could use

Object storage is particularly useful for archiving data, as it typically provides a convenient, accessible method of storing data which may need to be shared with a wide group of individuals.

Using alces storage commands

Your Flight Compute cluster includes command-line tools which can be used to enable access to existing **AWS S3** and **Dropbox** accounts. A Ceph storage platform with a compatible **RADOS-gateway** can also be accessed using S3 support. To enable access to these services, users must first enable them with the following commands:

- `alces storage enable s3` - enables **AWS S3** service
- `alces storage enable dropbox` - enables **Dropbox** service

Once enabled, a user can configure one or more storage services for use on the command-line, giving each one a friendly name to identify it. The syntax of the command is shown below:

```
alces storage configure <friendly-name> <type-of-storage>
```

For example; to configure access to an AWS S3 account using the access and secret key, the following commands can be used:

```
[alces@login1(scooby) ~]$ alces storage configure my-s3area1 s3
Display name [my-s3area1]:
Access key: PZHAA6I2OEDF9F2RQS8Q
Secret key: ********************
Service address [s3.amazonaws.com]:
alces storage configure: storage configuration complete
```

Note: If using a Ceph filesystem with a RADOS-gateway, enter the hostname of your gateway service as the `Service address` configuration item. For Amazon S3 based storage, choose the default service address.

When configuring a Dropbox account, the user is provided with a URL that must be copied and pasted into a browser session on their local client machine:

```
[alces@login1(scooby) ~]$ alces storage configure mydb dropbox
Display name [mydb]:
Please visit the following URL in your browser and click 'Authorize':

  https://www.dropbox.com/1/oauth/authorize?oauth_token=bdD4e2V2rjTf752u

Once you have completed authorization, please press ENTER to continue...
```

Copy the URL provided into your browser on your client system - you will be prompted to login to Dropbox (if you don't already have a session); click on the *Authorize* button on the next screen to allow your Flight Compute cluster to access the files stored in your Dropbox account.

Once you have set up one or more configurations, you can switch between the different storage spaces using the following commands:

```
[alces@login1(scooby) ~]$ alces storage use my-s3area1
alces storage use: storage configuration 'my-s3area1' now set as default
```

From the command-line, users can upload and download data from their configured storage areas. To upload data to an object storage area, use the `alces storage put <local-file> <object-name>` command; e.g.

```
[alces@login1(scooby) ~]$ alces storage put mydatafile datafile-may2016
alces storage put: mydatafile -> datafile-may2016

[alces@login1(scooby) ~]$ alces storage ls
2012-08-23 17:08        DIR   Public
2016-05-14 16:10       1335   datafile-may2016
```

```
2012-08-23 17:08     246000   Getting Started.pdf

[alces@login1(scooby) ~]$
```

To download data from an object storage service, use the `alces storage get <object-name> <local-file>` command; e.g.

```
[alces@login1(scooby) ~]$ alces storage get "Getting Started.pdf" instructions.pdf
alces storage get: Getting Started.pdf -> /home/alces/instructions.pdf

[alces@login1(scooby) ~]$ file instructions.pdf
instructions.pdf: PDF document, version 1.4

[alces@login1(scooby) ~]$
```

Users can also create new buckets in their object-storage service using the `alces storage mb <bucket-name>` command, and then put objects into the new bucket; e.g.

```
[alces@login1(scooby) data]$ alces storage mb newdata
alces storage mkbucket: created bucket newdata

[alces@login1(scooby) data]$ alces storage put datafile2 newdata/datafile2
alces storage put: datafile2 -> newdata/datafile2

[alces@login1(scooby) data]$ alces storage ls newdata
2016-05-14 16:14   20971520   datafile2

[alces@login1(scooby) data]$
```

Users can also recursively transfer entire buckets (including any buckets contained within) using the `-r` option to the `alces storage` command; e.g.

```
[alces@login1(scooby) ~]$ alces storage put -r datadir datadir2
alces storage put: datadir/datafile2 -> datadir2/datafile2
alces storage put: datadir/datafile3 -> datadir2/datafile3
alces storage put: datadir/datafile4 -> datadir2/datafile4
alces storage put: datadir/datafile5 -> datadir2/datafile5
alces storage put: datadir/datafile6 -> datadir2/datafile6

[alces@login1(scooby) ~]$
```

Saving data before terminating your cluster

When you've finished working with your Alces Flight Compute cluster, you can select to terminate it in the console for your Cloud service. This will stop any running instances and wipe the shared storage area before returning the block storage volumes back to the provider. Before you shutdown your cluster, users must ensure that they store their data safely in a persistent service, using one of the methods described in this documentation. When you next launch a Flight Compute cluster, you can restore your data from the storage service to begin processing again.

Graphical desktop access to your login node

Your Alces Flight Compute login node can also run graphical desktop sessions to support users who want to run interactive applications across the cluster. 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.

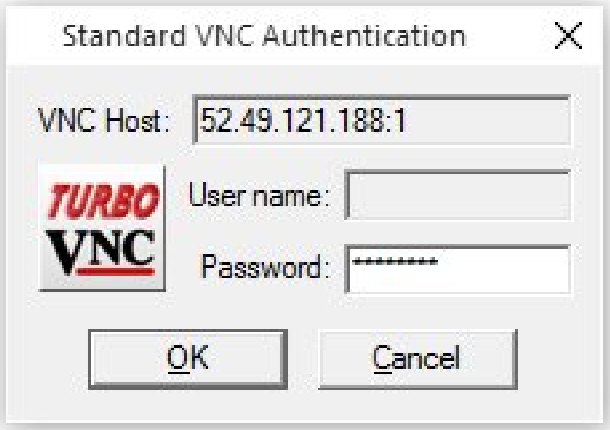
Launching a desktop session

All Flight Compute clusters come pre-installed with a Gnome desktop environment which users can start from the command-line as required. Users can launch a new session by using the `alces session start gnome` command. After launching the desktop, a message will be printed with connection details to access the new session:

```
VNC server started:
    Identity: b7d8e878-19b7-11e6-96cc-0a949a3e07d9
        Type: gnome
        Host: 52.49.121.188
        Port: 5901
     Display: 1
    Password: JqQJWkA5
   Websocket: 41361

Depending on your client, you can connect to the session using:

  vnc://alces:JqQJWkA5@52.49.121.188:5901
  52.49.121.188:5901
  52.49.121.188:1

If prompted, you should supply the following password: JqQJWkA5
```

Users need a VNC client to connect to the graphical desktop session - for a list of tested clients, see see *What is Alces Flight Compute?*.

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://alces:JqQJWkA5@52.49.121.188:5901`. 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 `52.49.121.188:5901` into their VNC client:


Standard VNC Authentication

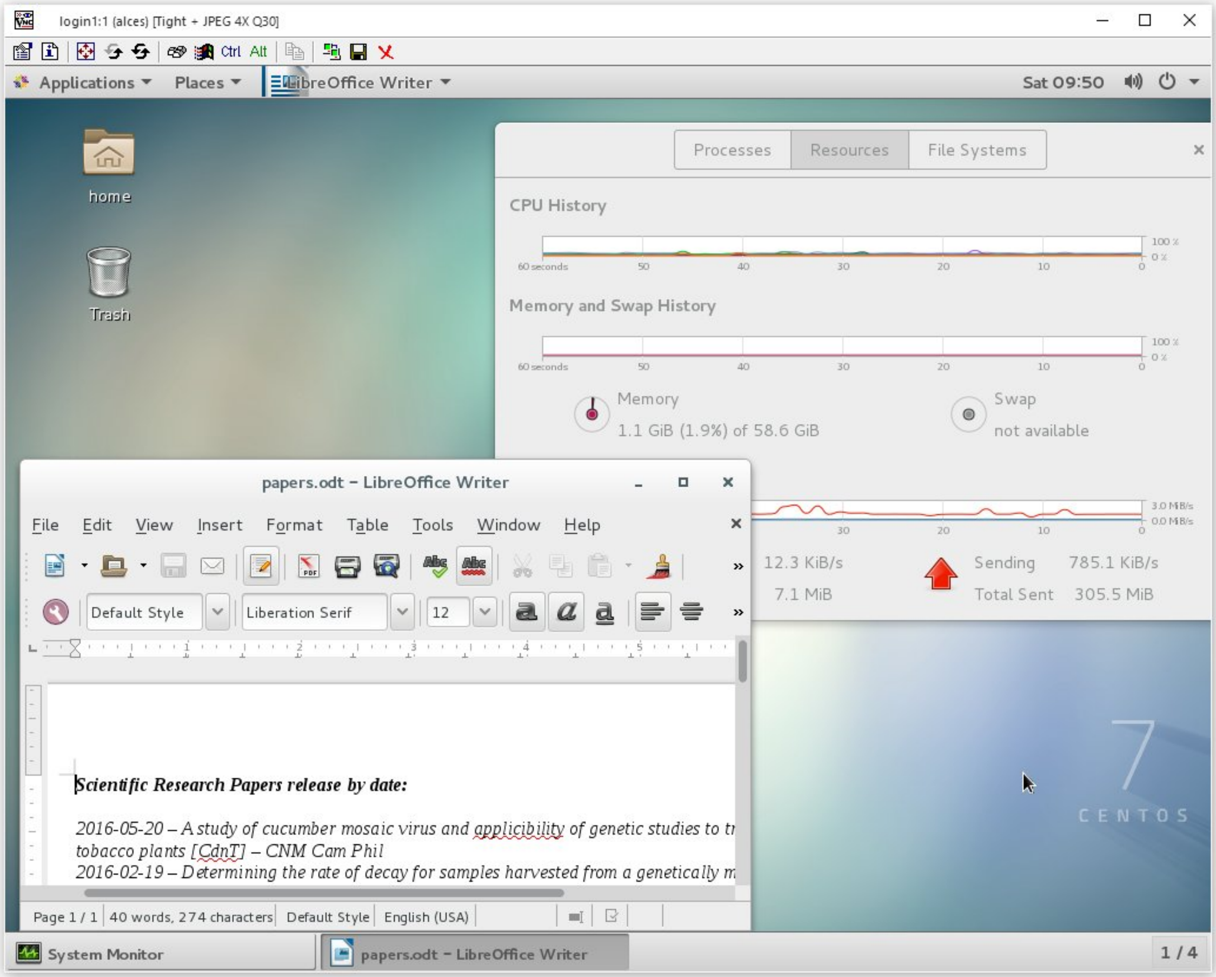
VNC Host: 52.49.121.188:1

User name:

Password: *********

A one-time randomized password is automatically generated automatically by Flight 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 environment as they would a local Linux machine:



Scientific Research Papers release by date:

2016-05-20 – A study of cucumber mosaic virus and applicibility of genetic studies to tr
tobacco plants [CdnT] – CNM Cam Phil
2016-02-19 – Determining the rate of decay for samples harvested from a genetically m

Connecting multiple users to the same session

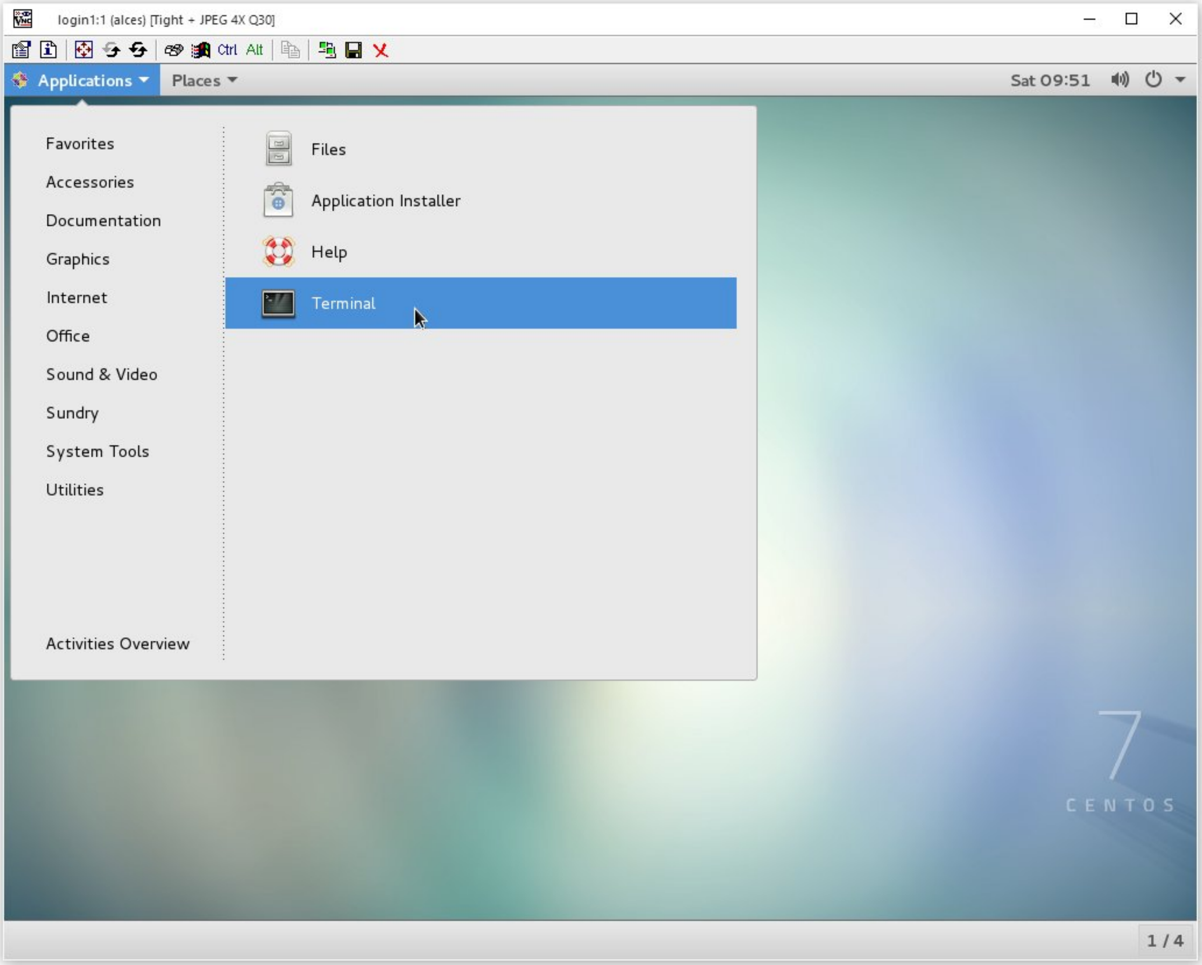
New graphical sessions are created in multi-user mode by default, allowing users from different locations to connect to the same session for training or collaborative projects. All users connect using the same connection details (e.g. IP-address and port number), and use the same one-time session password.

Please note that users are only permitted to connect to your Flight Compute cluster login node if their IP address is within the set of networks allowed in the `CIDR` setting made at launch time. If you have issues with secondary users connecting to a graphical desktop session, please try entering `0.0.0.0/0` as your CIDR at cluster launch time to allow access from all users.

Resizing the desktop to fit your screen

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:

```
[alces@login1(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
   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.

Using alces session commands to enable other types of session

Your Alces Flight Compute cluster can also support other types of graphical session designed to provide interactive applications directly to users. To view the available types of session, use the command `alces session avail`:

```
[alces@login1(scooby) ~]$ alces session avail
[ ] base/chrome
[ ] base/cinnamon
[*] base/default
[ ] base/fvwm
[*] base/gnome
[ ] base/icewm
[ ] base/terminal
[ ] base/trinity
[ ] base/xfce
```

Application types that are not marked with a star (*) need to be enabled before they can be started. To enable a new session type, use the command `alces session enable <type>`. Enabling a new session type will automatically install any required application and support files. Once enabled, users can start a new session using the command `alces session start <type>`.

Viewing and terminating running sessions

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

```
[alces@login1(scooby) ~]$ alces session list
+----------+------------+----------------------+-----------------+---------+------+---
↪-------+
| Identity | Type       | Host name            | Host address    | Display | Port |␣
↪Password |
+----------+------------+----------------------+-----------------+---------+------+---
↪-------+
| b7d8e878 | gnome      | login1               | 52.49.121.188   |      :1 | 5901 |␣
↪JqQJWkA5 |
| ce4c4372 | cinnamon   | login1               | 52.49.121.188   |      :2 | 5902 |␣
↪V9r2IuXb |
| d1d8342e | gnome      | login1               | 52.49.121.188   |      :3 | 5903 |␣
↪1HJRftxP |
| d4c69a18 | terminal   | login1               | 52.49.121.188   |      :4 | 5904 |␣
↪0du74LNn |
| d6d5f7cc | chrome     | login1               | 52.49.121.188   |      :5 | 5905 |␣
↪YbR8vkFy |
+----------+------------+----------------------+-----------------+---------+------+---
↪-------+
```

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

```
[alces@login1(scooby) ~]$ alces session info b7d8e878
Identity:      b7d8e878-19b7-11e6-96cc-0a949a3e07d9
Type:          gnome
```

```
Host name:    login1
Host address: 52.49.121.188
Port:         5901
Display:      1
Password:     JqQJWkA5
Websocket:    41361
URL:          vnc://alces:JqQJWkA5@52.49.121.188:5901
```

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 `alces session kill <session-ID>` command. A terminated session will be immediately stopped, disconnecting any users.

Software Applications

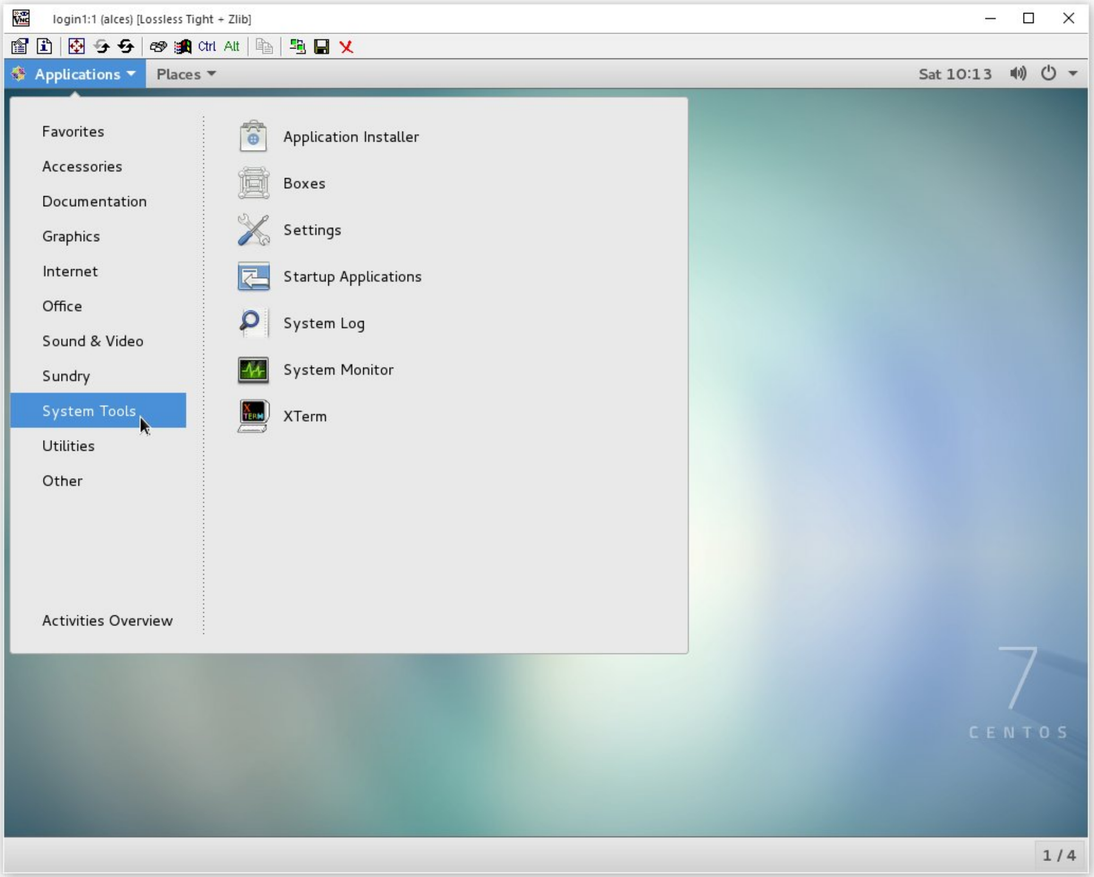
Flight Compute operating system

The current revision of Alces Flight Compute builds personal, ephemeral clusters based on a 64-bit CentOS 7.2 Linux distribution. The same operating system is installed on all login and compute nodes across the cluster. The Linux distribution includes a range of software tools and utilities, packaged by the CentOS project as RPM files. These packages are available for users to install as required on login and compute nodes using the `yum` command. You can also install other RPM packages on your Flight Compute cluster by copying them and installing them using the RPM command.

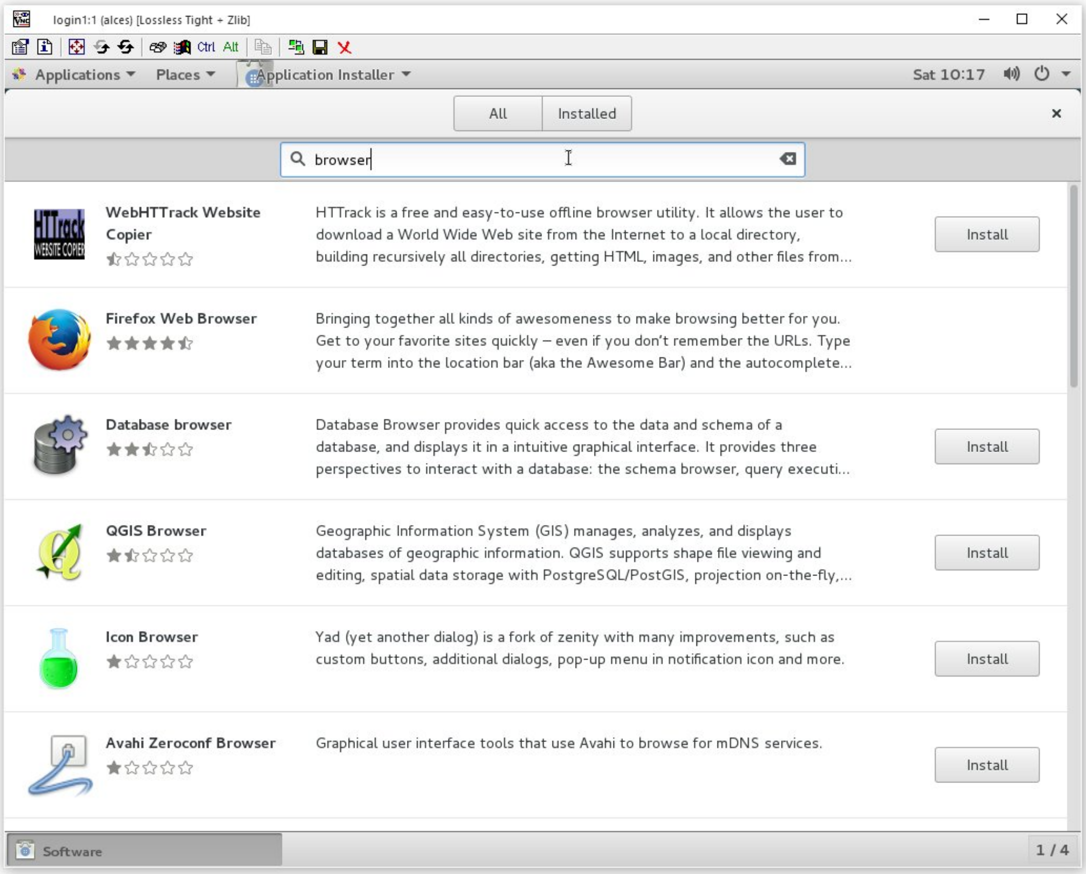
Installing Linux applications on the login node

Users who are new to Linux may find it convenient to install applications on the login node only via a graphical session from the Linux application catalogue. Follow the steps below to enable the graphical Linux package installer:

- **Start a new graphical session if you don't already have one; e.g.** `alces session start gnome`
- **Set a password for your user, using the sudo command. For example, if your Flight Compute cluster was launched w**
  `sudo passwd jane`
- Connect to the session from your VNC client, using the details provided.
- Click on the **Applications** menu in the top-right-hand corner of the desktop session, navigate to **System Tools**, and select the **Application Installer** option.



- Choose the applications to be installed on your login node; enter your password when prompted



WebHTTrack Website Copier

HTTrack is a free and easy-to-use offline browser utility. It allows the user to download a World Wide Web site from the Internet to a local directory, building recursively all directories, getting HTML, images, and other files from...

Install

Firefox Web Browser

Bringing together all kinds of awesomeness to make browsing better for you. Get to your favorite sites quickly – even if you don't remember the URLs. Type your term into the location bar (aka the Awesome Bar) and the autocomplete...

Database browser

Database Browser provides quick access to the data and schema of a database, and displays it in a intuitive graphical interface. It provides three perspectives to interact with a database: the schema browser, query executi...

Install

QGIS Browser

Geographic Information System (GIS) manages, analyzes, and displays databases of geographic information. QGIS supports shape file viewing and editing, spatial data storage with PostgreSQL/PostGIS, projection on-the-fly,...

Install

Icon Browser

Yad (yet another dialog) is a fork of zenity with many improvements, such as custom buttons, additional dialogs, pop-up menu in notification icon and more.

Install

Avahi Zeroconf Browser

Graphical user interface tools that use Avahi to browse for mDNS services.

Install

Note: Applications installed via the Linux packager are installed on the cluster login node only. We recommend using the **Alces Gridware** packaging system for installing applications to be used across multiple cluster nodes.

Shared cluster applications

While RPM packages are useful for system packages, they are not designed to manage software applications that have complex dependencies, span multiple nodes or coexist with other, incompatible applications. Alces Flight Compute clusters include a mechanism to install and manage software applications on your cluster called **Alces Gridware**. Gridware packages are centrally installed on your shared cluster filesystem, making them available to all cluster login and compute nodes. New applications are installed with supporting **environment module** files, and can be dynamically optimised at installation time for specific environments.

Shared application storage

For Flight Compute clusters launched from AWS Marketplace, your applications are automatically stored in the shared cluster filesystem, making them available to all login and compute nodes across the cluster. There are three directories used to host applications on your Flight Compute cluster:

- `/opt/gridware/` - Applications managed by Alces Gridware utility
- `/opt/apps/` - An empty directory for user-installed applications
- `/opt/clusterware/` - Flight admin data used to keep your cluster running

Installing cluster applications

Alces Flight Compute clusters include access to the online **Gridware** repository of software applications. This catalogue includes over 750 application, library, compiler and MPI versions which can be installed and run on your cluster. Software is installed by selecting it from the catalogue, compiling it on the cluster login node along with any software package dependencies, and installing it in the shared cluster application storage space. Once installed, applications can be run on the login and compute nodes interactively, or as part of job-scripts submitted to the cluster scheduler.

Application catalogue structure

Software applications are listed in the Alces Gridware repository with the structure `status/type/name/version/variant`, which corresponds to:

- **status** - packages are listed in the **main** repository if tested/stable, and the **volatile** repository otherwise.
- **type** - packages are listed as **apps** (applications), **libs** (shared libraries), **compilers** or **mpi** (message-passing interface API software for parallel applications)
- **name** - the name of the software package
- **version** - the published version of the software package
- **variant** - used at installation time to record information about a particular instance of a package, e.g. libraries used to build against, or arbitrary tags describing an installation option

For example, a package listed as `main/apps/bowtie2/2.2.6` is version 2.2.6 of the Bowtie2 application, from the stable repository.

Finding an application to install

From the login node of your Alces Flight Compute cluster, use the command `alces gridware list` to view the available packages for installation. To search for a particular package, use the `alces gridware search <search-word>` command; e.g.

```
[alces@login1(scooby) ~]$ alces gridware search bowtie
main/apps/bowtie/1.1.0   main/apps/bowtie2/2.2.6  main/apps/tophat/2.1.0
```

Note: By default, only the `main` repository is enabled; please read the instructions below to enable and use packages from the `volatile` repository.

Installing a Gridware application

Use the command `alces gridware install <package-name>` to install a new package; e.g.

```
[alces@login1(scooby) ~]$ alces gridware install apps/memtester
Preparing to install main/apps/memtester/4.3.0
Installing main/apps/memtester/4.3.0

 > Preparing package sources
        Download --> memtester-4.3.0.tar.gz ... OK
          Verify --> memtester-4.3.0.tar.gz ... OK

 > Preparing for installation
           Mkdir ... OK (/var/cache/gridware/src/apps/memtester/4.3.0/gcc-4.8.5)
```

```
         Extract ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK (/opt/gridware/depots/b7e5f115/el7/pkg/apps/memtester/4.3.0/
↪gcc-4.8.5)
         Install ... OK
          Module ... OK

Installation complete.
[alces@login1(scooby) ~]$
```

Where more than one version of the requested application exists in the repository, users will be prompted for more information when attempting to install:

```
[alces@login1(scooby) ~]$ alces gridware install apps/samtools
More than one matching package found, please choose one of:
main/apps/samtools/0.1.18  main/apps/samtools/0.1.19  main/apps/samtools/1.3

[alces@login1(scooby) ~]$ alces gridware install apps/samtools/1.3
Preparing to install main/apps/samtools/1.3
Installing main/apps/samtools/1.3

 > Preparing package sources
        Download --> samtools-1.3.tar.bz2 ... OK
          Verify --> samtools-1.3.tar.bz2 ... OK

 > Preparing for installation
           Mkdir ... OK (/var/cache/gridware/src/apps/samtools/1.3/gcc-4.8.5)
         Extract ... OK
    Dependencies ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK (/opt/gridware/depots/b7e5f115/el7/pkg/apps/samtools/1.3/gcc-
↪4.8.5)
         Install ... OK
          Module ... OK
    Dependencies ... OK

Installation complete.
```

For more complex applications, Alces Gridware may need to additionally build other applications, libraries and MPIs to support the installation. Users will be prompted if multiple installations will be required to make the requested package available:

```
[alces@login1(scooby) ~]$ alces gridware install apps/R
Preparing to install main/apps/R/3.2.3

WARNING: Package requires the installation of the following:
  main/apps/cmake/3.5.2, main/libs/blas/3.6.0, main/libs/lapack/3.5.0

Install these dependencies first?

Proceed (Y/N)?
```

Modules environment management

The Modules environment management system allows simple configuration of a users' Linux environment across a HPC compute cluster. It allows multiple software applications to be installed together across a group of systems, even if the different applications are incompatible with each other. Modules can also provide basic dependency analysis and resolution for software, helping users to make sure that their applications run correctly. An Alces Flight Compute cluster user can use modules to access the application software they need for running their jobs.

Note: Environment modules are included with your Alces Flight Compute cluster for convenience - users are free to use standard Linux configuration methods to setup their environment variables if they prefer.

Environment modules work by configuring three existing Linux environment variables:

```
$PATH
$LD_LIBRARY_PATH
$MANPATH
```

By manipulating these variables, the modules system can application binaries in your path, ensure that compatible library files are in your library path, and setup manual pages for applications. A library of module files is included with your Flight Compute cluster, and is automatically managed by the **Alces Gridware** software packager.

Using environment modules

Users can view the available environment modules on their Alces Flight Compute cluster by using the `module avail` command:

```
[alces@login1(scooby) ~]$ module avail
 ---  /opt/gridware/benchmark/el7/etc/modules  ---
   apps/hpl/2.1/gcc-4.8.5+openmpi-1.8.5+atlas-3.10.2
   apps/imb/4.0/gcc-4.8.5+openmpi-1.8.5
   apps/iozone/3.420/gcc-4.8.5
   apps/memtester/4.3.0/gcc-4.8.5
   compilers/gcc/system
   libs/atlas/3.10.2/gcc-4.8.5
   libs/gcc/system
   mpi/openmpi/1.8.5/gcc-4.8.5
   null
 ---  /opt/gridware/local/el7/etc/modules  ---
   compilers/gcc/system
   libs/gcc/system
   null
 ---  /opt/clusterware/etc/modules  ---
   null
   services/aws
   services/gridscheduler
 ---  /opt/apps/etc/modules  ---
   null
```

To load a new module for the current session, use the `module load <module-name>` command; any dependant modules will also be loaded automatically:

```
[alces@login1(scooby) ~]$ module load apps/memtester
apps/memtester/4.3.0/gcc-4.8.5
 | -- libs/gcc/system
 |     * --> OK
 |
OK
```

Note: Module names will auto-complete if you type the first few letters, then press the **<TAB>** button on your keyboard.

To unload a module file for the current session, use the `module unload <module name>` command. To allow users to configure specific versions of applications, the `module unload` command does not perform dependency analysis.

```
[alces@login1(scooby) ~]$ module unload apps/memtester
          apps/memtester/4.3.0/gcc-4.8.5 ... UNLOADING --> OK
```

Module files can be loaded interactively at the command-line or graphical desktop on both login and compute nodes in your cluster. They can also be loaded as part of a job-script submitted to the cluster job-scheduler.

Application specific variables

As well as the default environment variables (`$PATH, $LD_LIBRARY_PATH, $MANPATH`), modules included with Alces Flight Compute clusters also provide a number of additional Linux environment variables which are specific to the application being loaded. For example, to help users locate the application installation directory, the following variables are set automatically after laoding a named module file:

- **`{APP-NAME}DIR` - the location of the base application directory** e.g. for the **HPL** application, the variable `$HPLDIR` contains the base location of the HPL application
- **`{APP-NAME}BIN` - the location of the application directory holding executable binaries** e.g. for the **HPL** application, the variable `$HPLBIN` contains the location of binary files for HPL
- **`{APP-NAME}EXAMPLES` - the location of example files packaged with the application** e.g. for the **HPL** application, the variable `$HPLEXAMPLES` contains an example HPL.dat file

You can use the `module display <module-name>` command to view all the environment variables that will be created when loading the module file for an application.

Viewing application license information

The open-source community forms the life-blood of computer-aided scientific research across the world, with software developers creating and publishing their work for free in order to help others. This collaborative model relies on the kindness and dedication of individuals, public and private organisations and independent research groups in taking the time to develop and publish their software for the benefit of us all. Users of open-source software have a responsibility to obey the licensing terms, credit the original authors and follow their shining example by contributing back to the community where possible - either in the form of new software, feedback and bug-reports for the packages you use and highlighting software usage in your research papers and publications.

Applications installed by your Alces Flight Compute cluster include a module file that details the license type and original source URL for the package. Use the `alces display <module-name>` command to view this information:

```
[alces@login1(scooby) ~]$ module display apps/hpl
-------------------------------------------------------------------
/opt/gridware/benchmark/el7/etc/modules/apps/hpl/2.1/gcc-4.8.5+openmpi-1.8.5+atlas-3.
↪10.2:

```

```
module-whatis

            Title: HPL
          Summary: A Portable Implementation of the High-Performance Linpack␣
↪Benchmark for Distributed-Memory Computers
          License: Modified Free http://www.netlib.org/benchmark/hpl/copyright.html
            Group: Benchmarks
              URL: http://www.netlib.org/benchmark/hpl/

             Name: hpl
          Version: 2.1
           Module: apps/hpl/2.1/gcc-4.8.5+openmpi-1.8.5+atlas-3.10.2
      Module path: /opt/gridware/depots/1a995914/el7/etc/modules/apps/hpl/2.1/gcc-4.8.
↪5+openmpi-1.8.5+atlas-3.10.2
     Package path: /opt/gridware/depots/1a995914/el7/pkg/apps/hpl/2.1/gcc-4.8.
↪5+openmpi-1.8.5+atlas-3.10.2

       Repository: git+https://github.com/alces-software/packager-base.git@unknown
          Package: apps/hpl/2.1@9839698b
      Last update: 2016-05-05

          Builder: root@9bc1b720b60a
       Build date: 2016-05-05T17:16:55
    Build modules: mpi/openmpi/1.8.5/gcc-4.8.5, libs/atlas/3.10.2/gcc-4.8.5
         Compiler: compilers/gcc/system
           System: Linux 3.19.0-30-generic x86_64
             Arch: Intel(R) Xeon(R) CPU @ 2.30GHz, 1x1 (29028551)
     Dependencies: libs/gcc/system (using: libs/gcc/system)
                   mpi/openmpi/1.8.5/gcc-4.8.5 (using: mpi/openmpi/1.8.5/gcc-4.8.5)

For further information, execute:
    module help apps/hpl/2.1/gcc-4.8.5+openmpi-1.8.5+atlas-3.10.2

-------------------------------------------------------------------
```

Note: Please remember to credit open-source contributors by providing a URL to the supporting project along with your research papers and publications.

Configuring modules for your default session

The `module load` command configures your current session only - when a user logs out of the cluster or starts a new session, they are returned to their initial set of modules. This is often preferable for users wanting to include `module load` commands in their cluster job-scripts, but it is also possible to instruct environment modules to configure the default login environment so modules are automatically loaded at every login.

Use the `module initadd <module-file>` command to add a software package to the list of automatically loaded modules. The `module initrm <module-file` command will remove an application from the list of automatically loaded modules; the `module initlist` command will display what applications are currently set to automatically load on login.

Note: Commands to submit jobs to your cluster job-scheduler are automatically included in your users' **$PATH** via a `services/` module. If you unload this module or remove it from your list of automatically-loaded modules, you may not be able to submit jobs to the cluster scheduler.

Volatile Gridware repositories

Applications packaged in the `main` repository are tested to install without user interaction on an Alces Flight Compute cluster. For access to a larger catalogue of software, users can additionally enable the `volatile` software repository. Once enabled, advanced users can access the full list of available applications by choosing software along with any dependencies to install from the combined package list.

To enable volatile repositories, edit the `/opt/gridware/etc/gridware.yml` YAML file and un-comment the volatile repository by removing the # symbol at the start of line 11. Alternatively, users can enable the repository by using the following command:

```
sed -i 's?^# - /opt/clusterware/var/lib/gridware/repos/volatile? - /opt/clusterware/
↪var/lib/gridware/repos/volatile?g' /opt/gridware/etc/gridware.yml
```

Finally, run the `alces gridware update` command to refresh the application catalogue.

When installing packages from the volatile repo, users must resolve any dependencies before applications can be successfully installed. The Gridware packager will report any issues when attempting to install software from the volatile repo. The example below shows installation of the "beast" bioinformatics tool, which requires a Java Development Kit (JDK) to build:

```
[alces@login1(scooby) ~]$ alces gridware install volatile/apps/beast/1.7.5
Preparing to install volatile/apps/beast/1.7.5
Installing volatile/apps/beast/1.7.5

 > Preparing package sources
        Download --> beast-1.7.5.tgz ... OK
          Verify --> beast-1.7.5.tgz ... OK

 > Preparing for installation
           Mkdir ... OK (/var/cache/gridware/src/apps/beast/1.7.5/gcc-4.8.5)
         Extract ... OK

 > Proceeding with installation
         Compile ... ERROR: Package compilation failed

   Extract of compilation script error output:
   > In file included from NucleotideLikelihoodCore.c:2:0:
   > NucleotideLikelihoodCore.h:7:17: fatal error: jni.h: No such file or directory
   > #include <jni.h>
   > ^
   > compilation terminated.
   > make: *** [NucleotideLikelihoodCore.o] Error 1
[alces@login1(scooby) ~]$
```

The YUM utility can be used to identify any system packages which may satisfy build dependencies; e.g.

```
[alces@login1(scooby) ~]$ yum provides */jni.h
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.heanet.ie
 * extras: ftp.heanet.ie
 * updates: ftp.heanet.ie
extras/7/x86_64/filelists_db                                                     ␣
↪ | 296 kB  00:00:00
updates/7/x86_64/filelists_db                                                    ␣
↪ | 3.1 MB  00:00:00
1:java-1.6.0-openjdk-devel-1.6.0.36-1.13.8.1.el7_1.x86_64 : OpenJDK Development␣
↪Environment
```

```
Repo        : base
Matched from:
Filename    : /usr/lib/jvm/java-1.6.0-openjdk-1.6.0.36.x86_64/include/jni.h

[alces@login1(scooby) ~]$
```

Installing any dependencies may allow the software application to be installed as desired; e.g.

```
[alces@login1(scooby) ~]$ pdsh -g cluster 'sudo yum -y -e0 install java-1.8.0-openjdk-
↪devel'
Resolving Dependencies
--> Running transaction check
---> Package java-1.8.0-openjdk-devel.x86_64 1:1.8.0.91-0.b14.el7_2 will be installed
--> Processing Dependency: java-1.8.0-openjdk = 1:1.8.0.91-0.b14.el7_2 for package:␣
↪1:java-1.8.0-openjdk-devel-1.8.0.91-0.b14.el7_2.x86_64
--> Processing Dependency: libawt_xawt.so(SUNWprivate_1.1)(64bit) for package: 1:java-
↪1.8.0-openjdk-devel-1.8.0.91-0.b14.el7_2.x86_64
--> Processing Dependency: libawt_xawt.so()(64bit) for package: 1:java-1.8.0-openjdk-
↪devel-1.8.0.91-0.b14.el7_2.x86_64
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================
 Package                            Arch             Version                            ␣
↪Repository         Size
==========================================================================================
Installing:
 java-1.8.0-openjdk-devel           x86_64           1:1.8.0.91-0.b14.el7_2             ␣
↪updates           9.7 M
Installing for dependencies:
 java-1.8.0-openjdk                 x86_64           1:1.8.0.91-0.b14.el7_2             ␣
↪updates           219 k
 ttmkfdir                           x86_64           3.0.9-42.el7                       ␣
↪base               48 k
 xorg-x11-fonts-Type1               noarch           7.5-9.el7                          ␣
↪base              521 k

Transaction Summary
==========================================================================================
Install  1 Package (+3 Dependent packages)

Total download size: 11 M
Installed size: 42 M
Is this ok [y/d/N]: y
Running transaction
Installed:
  java-1.8.0-openjdk-devel.x86_64 1:1.8.0.91-0.b14.el7_2

Dependency Installed:
  java-1.8.0-openjdk.x86_64 1:1.8.0.91-0.b14.el7_2              ttmkfdir.x86_64 0:3.
↪0.9-42.el7
  xorg-x11-fonts-Type1.noarch 0:7.5-9.el7

Complete!

[alces@login1(scooby) ~]$ alces gridware install volatile/apps/beast/1.7.5
Preparing to install volatile/apps/beast/1.7.5
```

```
Installing volatile/apps/beast/1.7.5

WARNING: Build directory already exists:
  /var/cache/gridware/src/apps/beast/1.7.5/gcc-4.8.5

Proceed with a clean?

Proceed (Y/N)? y
           Clean ... OK

 > Preparing package sources
        Download --> beast-1.7.5.tgz ... SKIP (Existing source file detected)
          Verify --> beast-1.7.5.tgz ... OK

 > Preparing for installation
           Mkdir ... OK (/var/cache/gridware/src/apps/beast/1.7.5/gcc-4.8.5)
         Extract ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK (/opt/gridware/depots/b7e5f115/el7/pkg/apps/beast/1.7.5/gcc-4.
↪8.5)
         Install ... OK
          Module ... OK

Installation complete.
```

Installing packages from binary depots

Alces Flight Compute clusters also support separate application depots which are preconfigured to include specific suites of applications for particular purposes. Depots can be used for the following purposes:

- Creating a set of applications for a particular purpose (e.g. Bioinformatics, Engineering or Chemistry applications)
- Reducing software installation time by compiling in advance for a particular CPU type
- Collecting optimised applications together; e.g. those built with specialist accelerated compilers
- Packaging your frequently used applications in a convenient bundle
- Distributing your commercial applications (as permissible under the terms of the appropriate software license)

To list the available depots for your environment, use the command `alces gridware depot list`. New depots can be installed using the `alces gridware depot install <depot-name>` command; e.g.

```
[alces@login1(scooby) ~]$ alces gridware depot install benchmark
Installing depot: benchmark

 > Initializing depot: benchmark
      Initialize ... OK

Importing mpi-openmpi-1.8.5-el7.tar.gz

 > Fetching archive
        Download ... SKIP (Existing source file detected)
 > Preparing import
         Extract ... OK
          Verify ... OK

 > Processing mpi/openmpi/1.8.5/gcc-4.8.5
       Preparing ... OK
       Importing ... OK
     Permissions ... OK

 > Finalizing import
          Update ... OK
    Dependencies ... OK

Importing libs-atlas-3.10.2-el7.tar.gz

 > Fetching archive
        Download ... SKIP (Existing source file detected)

 > Preparing import
         Extract ... OK
          Verify ... OK

 > Processing libs/atlas/3.10.2/gcc-4.8.5
       Preparing ... OK
       Importing ... OK
     Permissions ... OK

 > Finalizing import
          Update ... OK
    Dependencies ... OK

 [alces@login1(scooby) ~]$
```

Once installed, enable a new depot using the `alces gridware depot enable <depot-name>` command; e.g.

```
[alces@login1(scooby) ~]$ alces gridware depot enable benchmark

 > Enabling depot: benchmark
          Enable ... OK
```

Requesting new applications in Gridware

The list of applications available in the Gridware repository expands over time as more software is added and tested on Flight Compute clusters. Wherever possible, software is not removed from the repository, allowing users to rely on applications continuing to be available for a particular release of Alces Flight. New versions of existing applications are also added over time - newly launched Flight Compute clusters automatically use the latest revision of the Gridware repository; use the `alces gridware update` command to refresh any running Flight Compute clusters with the latest updates.

If you need to use an application that isn't already part of the Alces Gridware project, there are three methods you can use to get access to the application:

1. Install the application yourself manually (see below). This is a good first step for any new software package, as it will allow you to evaluate its use on a cluster and confirm that it works as expected in a Flight Compute

cluster environment.

2. Request the addition of an application via the community support site. Please include as much information about the application as possible in your request to help new users of the package. There is no fee for requesting software via the community support site - this service is provided to benefit users worldwide by providing convenient access to the best open-source software packages available.

3. If you have an urgent need for a new software package, users can fund consultancy time to have packages added to Gridware repository. Please add details of your funding offer to your enhancement request ticket on the community support site, and a software engineer will contact you with more details.

Manually installing applications on your cluster

Your Alces Flight Compute cluster also allows manual installation of software applications into the `/opt/apps/` directory. This is useful for commercial applications that you purchase, and for software which you've written yourself or at your business or institution. Your Flight Compute cluster runs standard CentOS7, and should be compatible with any application tested on a CentOS, Scientific Linux or RedHat Enterprise Linux 7 distribution. It is often possible to run applications designed to run on other distributions with minimal modifications.

Install new applications into a sub-directory of the `/opt/apps/` directory - this location is available on both login and compute nodes, allowing software to be run across your cluster. A example environment module tree is also included for use with manually installed applications - add new modules into the `/opt/apps/etc/modules/` directory to be included here. Documentation on creating your own module files is available here.

Parallel (MPI) Applications

What is an MPI?

High-performance compute clusters are often used to run *Parallel Applications* - i.e. software applications which simultaneously use resources from two or more computers at the same time. This can allow software programs to run bigger jobs, run them faster, and to work with larger data-sets than can be processed on a single computer. Parallel programming is hard - developing software to run on a single computer is difficult enough, but extending applications to run across multiple computers at the same means doing many more internal checks while your program is running to make sure your software runs correctly, and to deal with any errors that occur.

A number of standards for parallel programming have been produced to assist software developers in this task. These published standards are often accompanied by an implementation of a software application programming interface (API) which presents a number of standard methods for parallel communication to software developers. By writing their software to be compatible with a published API, software developers can save time by relying on the API to deal with the parallel communications themselves (e.g. transmitting messages, dealing with errors and timeouts, interfacing with different hardware, etc.). The APIs for parallel processing are commonly known as message-passing interfaces (MPIs).

What MPIs are available?

A number of different MPIs are under active development; for Linux clusters, there are a number of common versions available, including:

- OpenMPI; a modern, open-source implementation supporting a wide array of hardware, Linux distributions and applications
- MPICH; an older open-source implementation largely superceded by OpenMPI, but still available for compatibility reasons

- MVAPICH; an open-source MPI supporting verbs transport across Infiniband fabrics
- Intel MPI; a commercial MPI optimised for Intel CPUs and interconnects
- IBM Platform MPI; a commercial MPI optimised for particular commercial applications and interconnects

The choice of which MPI to use for any particular use-case can depend on the application you want to run, the hardware you have available to run it on, if you have a license for a commercial application, and many other factors. Discussion and comparison of the available MPIs is outside the scope of this documentation - however, it should be possible to install and run any application that supports your underlying platform type and Linux distribution on an Alces Flight Compute cluster.

How do I use an MPI?

Most MPIs are distributed as a collection of:

- Software libraries that your application is compiled against
- Utilities to launch and manage an MPI session
- Documentation and integrations with application and scheduler software

You can use your Alces Flight Compute cluster to install the MPI you want to use, then compile and install the software application to be run on the cluster. Alternatively, users can install their own MPI and application software manually into the `/opt/apps/` directory of the cluster.

To run a parallel application, users typically start a new MPI session with parameters which instruct the MPI which nodes to include in the job, and which application to run. Each MPI requires parameters to be specified in the correct syntax - most also require a list of the compute nodes that will be participating in the job to be provided when a new session is started.

Running an MPI job via the cluster scheduler

Most users utilise the cluster job-scheduler to orchestrate launching of parallel jobs. The job-scheduler is responsible for identifying which nodes will be participating in the parallel job, and passing that information on to the MPI. When an MPI is installed on your Alces Flight Compute cluster using the `alces gridware` command, an integration for your chosen job-scheduler is automatically installed and configured at the same time. Please see the next section of this documentation for more information on launching a parallel job via your cluster job-scheduler.

Running an MPI job manually

In some environments, users may wish to manually run MPI jobs across the compute nodes in their cluster without using the job-scheduler. This can be useful when writing and debugging parallel applications, or when running parallel applications which launch directly on compute nodes without requiring a scheduler. A number of commercial applications may fall into this category, including Ansys Workbench, Ansys Fluent, Mathworks Matlab and parallelised R-jobs.

Note: Before running applications manually on compute nodes, verify that auto-scaling of your cluster is not enabled. Auto-scaling typically uses job-scheduler information to control how many nodes are available in your cluster, and should be disabled if running applications manually. Use the command `alces configure autoscaling disable` to turn off autoscaling before attempting to run jobs manually on your cluster compute nodes.

The example below demonstrates how to manually run the **Intel Message-passing Benchmark** application through **OpenMPI** on an Alces Flight Compute cluster. The exact syntax for your application and MPI may vary, but users should be able to follow the concepts discussed below to run their own software. You will need at least two compute nodes available to run the following example.

1. Install the application and MPI you want to run. The **benchmarks** software depot includes both **OpenMPI** and **IMB** applications, so install and enable that by running these commands:
   - `alces gridware depot install benchmark`
   - `alces gridware depot enable benchmark`
2. Create a list of compute nodes to run the job on. The following command will use your **genders** group to create a hostfile with one hostname per line:
   - `cd ; nodeattr -n nodes > mynodesfile`
3. Load the module file for the **IMB** application; this will also load the **OpenMPI** module file as a dependency. Add the module file to load automatically at login time:
   - `module initadd apps/imb`
   - `module load apps/imb`
4. Start the parallel application in a new **mpirun** session, with the following parameters:
   - `-np 2` - use two CPU cores in total
   - `-npernode 1` - place a maximum of one MPI thread on each node
   - `-hostfile mynodesfile` - use the list of compute nodes defined in the file `mynodesfile` for the MPI job (as generated in step 2 above)
   - `$IMBBIN/IMB-MPI1` - run the binary **IMB-MPI1**, located in the `$IMBBIN` directory configured by the `apps/imb` module
   - `PingPong` - a parameter to the **IMB-MPI1** application, this option instructs it to measure the network bandwidth and latency between nodes

```
[alces@login1(scooby) ~]$ mpirun -np 2 -npernode 1 -hostfile mynodesfile $IMBBIN/IMB-
↪MPI1 PingPong

 benchmarks to run PingPong
#------------------------------------------------------------
#    Intel (R) MPI Benchmarks 4.0, MPI-1 part
#------------------------------------------------------------
# Date                  : Sat May 14 15:37:49 2016
# Machine               : x86_64
# System                : Linux
# Release               : 3.10.0-327.18.2.el7.x86_64
# Version               : #1 SMP Thu May 12 11:03:55 UTC 2016
# MPI Version           : 3.0
# MPI Thread Environment:

# Calling sequence was:
# /opt/gridware/depots/2fe5b915/el7/pkg/apps/imb/4.0/gcc-4.8.5+openmpi-1.8.5/bin//IMB-
↪MPI1 PingPong

# Minimum message length in bytes:   0
# Maximum message length in bytes:   4194304
#
# MPI_Datatype                   :   MPI_BYTE
# MPI_Datatype for reductions    :   MPI_FLOAT
# MPI_Op                         :   MPI_SUM
#
```

```
# List of Benchmarks to run:
# PingPong

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
            0         1000         3.37         0.00
            1         1000         3.22         0.30
            2         1000         3.89         0.49
            4         1000         3.96         0.96
            8         1000         3.99         1.91
           16         1000         3.87         3.95
           32         1000         3.90         7.83
           64         1000         3.91        15.59
          128         1000         4.62        26.44
          256         1000         4.86        50.19
          512         1000         5.89        82.95
         1024         1000         6.08       160.58
         2048         1000         6.98       279.72
         4096         1000        10.35       377.26
         8192         1000        17.43       448.32
        16384         1000        31.13       501.90
        32768         1000        56.90       549.22
        65536          640        62.37      1002.09
       131072          320       127.54       980.10
       262144          160       230.23      1085.88
       524288           80       413.88      1208.08
      1048576           40       824.77      1212.45
      2097152           20      1616.90      1236.93
      4194304           10      3211.40      1245.56

# All processes entering MPI_Finalize
```

Cluster job schedulers

What is a batch job scheduler?

Most existing High-performance Compute Clusters are managed by a job scheduler; also known as the batch scheduler, workload manager, queuing system or load-balancer. The scheduler allows multiple users to fairly share compute nodes, allowing system administrators to control how resources are made available to different groups of users. All schedulers are designed to perform the following functions:

- Allow users to submit new jobs to the cluster
- Allow users to monitor the state of their queued and running jobs
- Allow users and system administrators to control running jobs
- Monitor the status of managed resources including system load, memory available, etc.

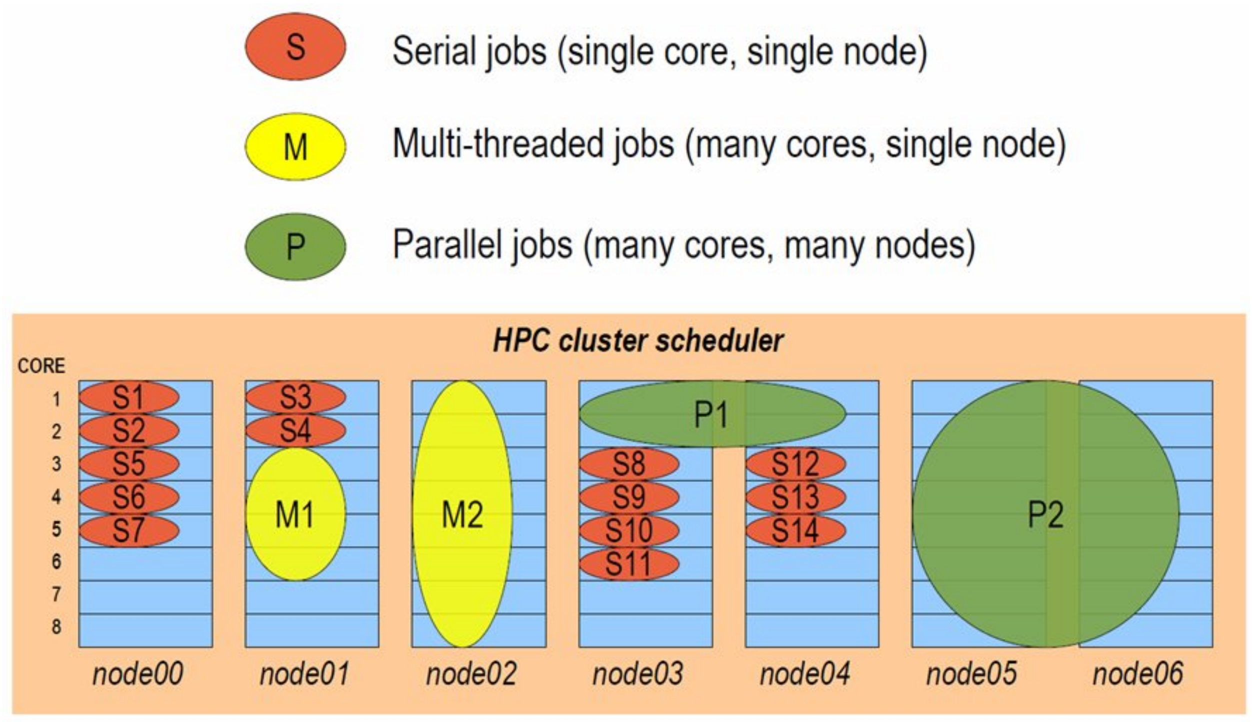
When a new job is submitted by a user, the cluster scheduler software assigns compute cores and memory to satisfy the job requirements. If suitable resources are not available to run the job, the scheduler adds the job to a queue until enough resources are available for the job to run. You can configure the scheduler to control how jobs are selected from the queue and executed on cluster nodes, including automatically preparing nodes to run parallel MPI jobs. Once a job has finished running, the scheduler returns the resources used by the job to the pool of free resources, ready to run another user job.

Types of compute job

Users can run a number of different types of job via the cluster scheduler, including:

- **Batch jobs**; single-threaded applications that run only on one compute core
- **Array jobs**; two or more similar batch jobs which are submitted together for convenience
- **SMP** or **multi-threaded** jobs; multi-threaded applications that run on two or more compute cores on the same compute node
- **Parallel jobs**; multi-threaded applications making use of an MPI library to run on multiple cores spread over one or more compute nodes

The cluster job-scheduler is responsible for finding compute nodes in your cluster to run all these different types of jobs on. It keeps track of the available resources and allocates jobs to individual groups of nodes, making sure not to over-commit CPU and memory. The example below shows how a job-scheduler might allocate jobs of different types to a group of 8-CPU-core compute nodes:


S
Serial jobs (single core, single node)
M
Multi-threaded jobs (many cores, single node)
P
Parallel jobs (many cores, many nodes)
HPC cluster scheduler
CORE
1
2
3
4
5
6
7
8
S1
S2
S5
S6
S7
S3
S4
M1
M2
P1
S8
S9
S10
S11
S12
S13
S14
P2
node00
node01
node02
node03
node04
node05
node06

Interactive and batch jobs

Users typically interact with compute clusters by running either **interactive** or **batch** (or **non-interactive**) jobs.

- An interactive job is one that the user directly controls, either via a graphical interface or by typing at the command-prompt.
- A batch job is run by writing a list of instructions that are passed to compute nodes to run at some point in the future.

Both methods of running jobs can be equally as efficient, particularly on a personal, ephemeral cluster. Both classes of job can be of any type - for example, it's possible to run interactive parallel jobs and batch multi-threaded jobs across your cluster. The choice of which class of job-type you want to use will depend on the application you're running, and which method is more convenient for you to use.

Why use a job-scheduler on a personal cluster?

Good question. On shared multi-user clusters, a job-scheduler is often used as a control mechanism to make sure that users don't unfairly monopolise the valuable compute resources. In extreme cases, the scheduler may be wielded by system administrators to force "good behaviour" in a shared environment, and can feel like an imposition to cluster users.

With your own personal cluster, you have the ability to directly control the resources available for your job - you don't need a job-scheduler to limit your usage.

However - there are a number of reasons why your own job-scheduler can still be a useful tool in your cluster:

1. It can help you organise multi-stage work flows, with batch jobs launching subsequent jobs in a defined process.
2. It can automate launching of MPI jobs, finding available nodes to run applications on.
3. It can help prevent accidentally over-allocating CPUs or memory, which could lead to nodes failing.
4. It can help bring discipline to the environment, providing a consistent method to replicate the running of jobs in different environments.
5. Jobs queued in the scheduler can be used to trigger scaling-up the size of your cluster, with compute nodes released from the cluster when there are no jobs to run, saving you money.

Your Alces Flight Compute cluster comes with a job-scheduler pre-installed, ready for you to start using. The scheduler uses very few resources when idle, so you can choose to use it if you find it useful, or run jobs manually across your cluster if you prefer.

Cluster job scheduler support

This version of Alces Flight Compute includes support for Open Grid Scheduler (OGS) - an open-source job scheduler, built from the codebase of what was originally the Sun Grid Engine (SGE) job scheduler. The syntax and usage of commands is identical to historical SGE syntax, and users can typically migrate from one to another with no issues.

It is also possible for users to download and install their own job-scheduler across their Flight Compute cluster - any product compatible with RedHat Enterprise Linux 7 and derivatives should be possible to run on your cluster.

We intend to include support for other job-schedulers in future versions of Alces Flight Compute - please let us know at the Alces Flight Community Support site if you have a particular favourite that you'd like us to include, and the reasons why.

Open Grid Scheduler (SGE)

The Open Grid Scheduler (OGS) cluster job-scheduler is an open-source implementation of the popular Sun Grid-Engine (SGE) codebase, with recent patches and updates to support newer Linux distributions. The syntax and usage of commands is identical to historical SGE syntax, and users can typically migrate from one to another with no issues. This documentation provides a guide to using OGS on your Alces Flight Compute cluster to run different types of jobs.

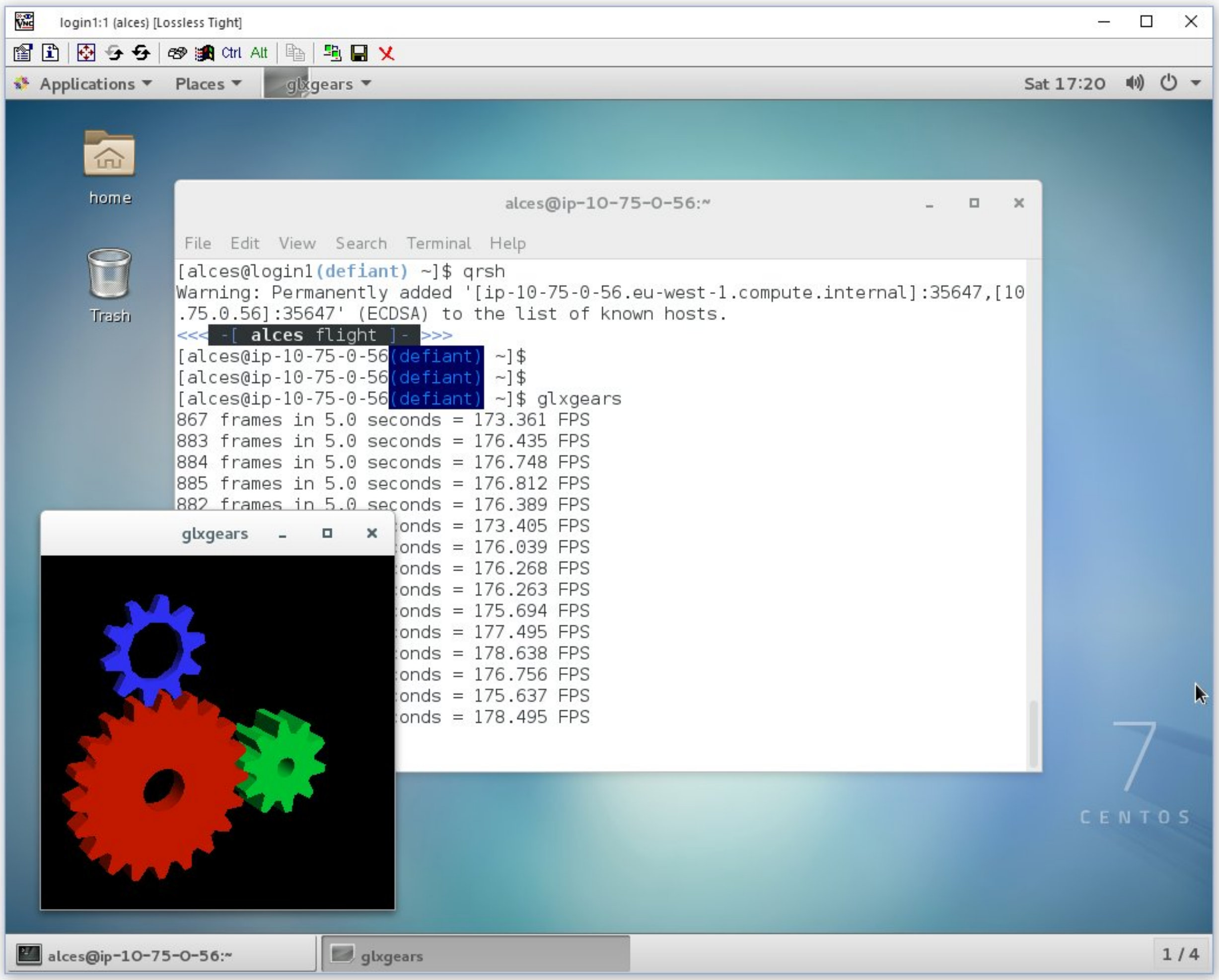
See *Cluster job schedulers* for a description of the different use-cases of a cluster job-scheduler.

Running an Interactive job

You can start a new interactive job on your Flight Compute cluster by using the `qrsh` command; the scheduler will search for an available compute node, and provide you with an interactive login shell on the node if one is available.

```
[alces@login1(scooby) ~]$ qrsh
Warning: Permanently added '[ip-10-75-0-21.eu-west-1.compute.internal]:53024,[10.75.0.
↪21]:53024' (ECDSA) to the list of known hosts.

<<< -[ alces flight ]- >>>
[alces@ip-10-75-0-21(scooby) ~]$ hostname -f
ip-10-75-0-21.eu-west-1.compute.internal

[alces@ip-10-75-0-21(scooby) ~]$ module load apps/R

[alces@ip-10-75-0-21(scooby) ~]$ R

R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
>
```

Alternatively, the `qrsh` command can also be executed from an interactive desktop session; the job-scheduler will automatically find an available compute node to launch the job on. Applications launched from within the qrsh session are executed on the assigned cluster compute node:



```
[alces@login1(defiant) ~]$ qrsh
Warning: Permanently added '[ip-10-75-0-56.eu-west-1.compute.internal]:35647,[10
.75.0.56]:35647' (ECDSA) to the list of known hosts.
<<< -[ alces flight ]- >>>
[alces@ip-10-75-0-56(defiant) ~]$
[alces@ip-10-75-0-56(defiant) ~]$
[alces@ip-10-75-0-56(defiant) ~]$ glxgears
867 frames in 5.0 seconds = 173.361 FPS
883 frames in 5.0 seconds = 176.435 FPS
884 frames in 5.0 seconds = 176.748 FPS
885 frames in 5.0 seconds = 176.812 FPS
882 frames in 5.0 seconds = 176.389 FPS
onds = 173.405 FPS
onds = 176.039 FPS
onds = 176.268 FPS
onds = 176.263 FPS
onds = 175.694 FPS
onds = 177.495 FPS
onds = 178.638 FPS
onds = 176.756 FPS
onds = 175.637 FPS
onds = 178.495 FPS
```

When you've finished running your application, simply type `logout`, or press **CTRL+D** to exit the interactive job.

If the job-scheduler could not satisfy the resources you've requested for your interactive job (e.g. all your available compute nodes are busy running other jobs), it will report back after a few seconds with an error:

```
[alces@login1(scooby) ~]$ qrsh
Your "qrsh" request could not be scheduled, try again later.
```

You can force an interactive job to queue for available resources by adding the parameter `-now no` to your `qrsh` command.

Submitting a batch job

Batch (or non-interactive) jobs allow users to leverage one of the main benefits of having a cluster scheduler; jobs can be queued up with instructions on how to run them and then executed across the cluster while the user does something else. Users submit jobs as scripts, which include instructions on how to run the job - the output of the job (*stdout* and *stderr* in Linux terminology) is written to a file on disk for review later on. You can write a batch job that does anything that can be typed on the command-line.

We'll start with a basic example - the following script is written in bash (the default Linux command-line interpreter). You can create the script yourself using the Nano command-line editor - use the command `nano simplejobscript.sh` to create a new file, then type in the contents below. The script does nothing more than print some messages to the screen (the **echo** lines), and sleeps for 120 seconds. We've saved the script to a file called `simplejobscript.sh` - the `.sh` extension helps to remind us that this is a *shell* script, but adding a filename extension isn't strictly necessary for Linux.

```
#!/bin/bash -l
echo "Starting running on host $HOSTNAME"
sleep 120
echo "Finished running - goodbye from $HOSTNAME"
```

Note: We use the "-l" option to bash on the first line of the script to request a login session. This ensures that environment modules can be loaded as required as part of your script.

We can execute that script directly on the login node by using the command `bash simplejobscript.sh` - after a couple of minutes, we get the following output:

```
Starting running on host login1
Finished running - goodbye from login1
```

To submit your jobscript to the cluster job scheduler, use the command `qsub simplejobscript.sh`. The job scheduler should immediately report the job-ID for your job; your job-ID is unique for your current Alces Flight Compute cluster - it will never be repeated once used.

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job 3 ("simplejobscript.sh") has been submitted

[alces@login1(scooby) ~]$
```

Viewing and controlling queued jobs

Once your job has been submitted, use the `qstat` command to view the status of the job queue. If you have available compute nodes, your job should be shown in `r` (running) state; if your compute nodes are busy, or you've launched an auto-scaling cluster and currently have no running nodes, your job may be shown in `qw` (queuing/waiting) state until compute nodes are available to run it.

```
[alces@login1(scooby) ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          ␣
↪            slots ja-task-ID
-----------------------------------------------------------------------------------------------
↪---------------------------
      3 11.02734 simplejobs alces        r     05/15/2016 09:32:54 byslot.q@ip-10-75-0-
↪131.eu-wes     1
```

You can keep running the `qstat` command until your job finishes running and disappears from the queue. The output of your batch job will be stored in a file for you to look at. The default location to store the output file is your home-directory - the output file will be named in the format `<jobscript-name>.o<job-ID>`. So - in the example above, our jobscript was called `simplejobscript.sh` and the job-ID was `3`, so our output file is located at `~/simplejobscript.sh.o3`. You can use the Linux `more` command to view your output file:

```
[alces@login1(scooby) ~]$ more ~/simplejobscript.sh.o3
Starting running on host ip-10-75-0-131.eu-west-1.compute.internal
Finished running - goodbye from ip-10-75-0-131.eu-west-1.compute.internal
```

Your job runs on whatever node the scheduler can find which is available for use - you can try submitting a bunch of jobs at the same time, and using the `qstat` command to see where they run. The scheduler is likely to spread them around over different nodes in your cluster (if you have multiple nodes). The login node is not included in your cluster for scheduling purposes - jobs submitted to the scheduler will only be run on your cluster compute nodes. You can use the `qdel <job-ID>` command to delete a job you've submitted, whether it's running or still in queued state.

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job 4 ("simplejobscript.sh") has been submitted
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job 5 ("simplejobscript.sh") has been submitted
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job 6 ("simplejobscript.sh") has been submitted
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job 7 ("simplejobscript.sh") has been submitted
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job 8 ("simplejobscript.sh") has been submitted
[alces@login1(scooby) ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          ␣
↪           slots ja-task-ID
-----------------------------------------------------------------------------------------------
↪----------------------------
      4 11.15234 simplejobs alces        r     05/15/2016 09:43:48 byslot.q@ip-10-75-
↪0-117.eu-wes     1
      5 11.02734 simplejobs alces        r     05/15/2016 09:43:49 byslot.q@ip-10-75-
↪0-126.eu-wes     1
      6 11.02734 simplejobs alces        r     05/15/2016 09:43:49 byslot.q@ip-10-75-
↪0-131.eu-wes     1
      7 11.02734 simplejobs alces        r     05/15/2016 09:43:49 byslot.q@ip-10-75-
↪0-154.eu-wes     1
      8 11.02734 simplejobs alces        r     05/15/2016 09:43:49 byslot.q@ip-10-75-
↪0-199.eu-wes     1

[alces@login1(scooby) ~]$ qdel 8
alces has registered the job 8 for deletion
```

Viewing compute host status

Users can use the `qhost` command to view the status of compute node hosts in your Flight Compute cluster.

```
[alces@login1(scooby) ~]$ qhost
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
ip-10-75-0-117          linux-x64      36  0.01   58.6G  602.7M    2.0G     0.0
ip-10-75-0-126          linux-x64      36  0.01   58.6G  593.6M    2.0G     0.0
ip-10-75-0-131          linux-x64      36  0.01   58.6G  601.9M    2.0G     0.0
ip-10-75-0-132          linux-x64      36  0.01   58.6G  589.5M    2.0G     0.0
ip-10-75-0-154          linux-x64      36  0.01   58.6G  603.7M    2.0G     0.0
ip-10-75-0-199          linux-x64      36  0.01   58.6G  604.9M    2.0G     0.0
ip-10-75-0-202          linux-x64      36  0.01   58.6G  591.4M    2.0G     0.0
ip-10-75-0-211          linux-x64      36  0.01   58.6G  586.8M    2.0G     0.0
```

The `qhost` output will show (from left-to-right):

- The hostname of your compute nodes
- The architecture of your compute nodes (typically 64-bit Linux for Flight Compute clusters)
- The detected number of CPUs (including hyper-threaded cores)

- The Linux run-queue length; e.g. for a 36-core node, a load of `36.0` indicates that the system is 100% loaded
- Memory statistics; the total and used amount of physical RAM and configured swap memory

Default resources

In order to promote efficient usage of your cluster, the job-scheduler automatically sets a number of default resources to your jobs when you submit them. These defaults must be overridden by users to help the scheduler understand how you want it to run your job - if we don't include any instructions to the scheduler, then our job will take the defaults shown below:

- Number of CPU cores for your job: `1`
- Maximum job runtime (in hours): `24`
- Output file location: `~/<jobscript-name>.o<jobID>`
- Output file style: `stdout` and `stderr` merged into a single file.
- **Amount of memory for your job: the arithmetic sum of** `total memory per node / total cores per node` e.g. with 36 core nodes that have 60GB of RAM, the default memory per job is set to around 1.5GB

This documentation will explain how to change these limits to suit the jobs that you want to run. You can also disable these limits if you prefer to control resource allocation manually by yourself.

Note: Scheduler limits are automatically enforced - e.g. if your job exceeds the requested runtime or memory allocation, it will automatically be stopped.

Providing job-scheduler instructions

Most cluster users will want to provide instructions to the job-scheduler to tell it how to run their jobs. The instructions you want to give will depend on what your job is going to do, but might include:

- Naming your job so you can find it again
- Controlling how job output files are written
- Controlling when your job will be run
- Requesting additional resources for your job

Job instructions can be provided in two ways; they are:

1. **On the command line**, as parameters to your `qsub` or `qrsh` command.

   e.g. you can set the name of your job using the `-N <name>` option:

```
[alces@login1(scooby) ~]$ qsub -N newname simplejobscript.sh
Your job 16 ("newname") has been submitted

[alces@login1(scooby) ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue            ␣
↪          slots ja-task-ID
-------------------------------------------------------------------------------------
↪---------------------------
     16 11.02734 newname    alces        r     05/15/2016 10:09:13 byslot.q@ip-10-75-
↪0-211.eu-wes      1
```

2. For batch jobs, job scheduler instructions can also **included in your job-script** on a line starting with the special identifier `#$`.

   e.g. the following job-script includes a `-N` instruction that sets the name of the job:

```
#!/bin/bash -l
#$ -N newname
echo "Starting running on host $HOSTNAME"
sleep 120
echo "Finished running - goodbye from $HOSTNAME"
```

Including job scheduler instructions in your job-scripts is often the most convenient method of working for batch jobs - follow the guidelines below for the best experience:

- Lines in your script that include job-scheduler instructions must start with `#$` at the beginning of the line
- You can have multiple lines starting with `#$` in your job-script, with normal scripts lines in-between.
- You can put multiple instructions separated by a space on a single line starting with `#$`
- The scheduler will parse the script from top to bottom and set instructions in order; if you set the same parameter twice, the second value will be used and a warning will be printed at submission time.
- Instructions provided as parameters to `qsub` override values specified in job-scripts.
- Instructions are parsed at job submission time, before the job itself has actually run. That means you can't, for example, tell the scheduler to put your job output in a directory that you create in the job-script itself - the directory will not exist when the job starts running, and your job will fail with an error.
- You can use dynamic variables in your instructions (see below)

Dynamic scheduler variables

Your cluster job scheduler automatically creates a number of pseudo environment variables which are available to your job-scripts when they are running on cluster compute nodes, along with standard Linux variables. Useful values include the following:

- `$HOME` The location of your home-directory
- `$USER` The Linux username of the submitting user
- `$HOSTNAME` The Linux hostname of the compute node running the job
- `$JOB_ID` The job-ID number for the job
- `$JOB_NAME` The configured job name
- `$SGE_TASK_ID` For task array jobs, this variable indicates the task number. This variable is not defined for non-task-array jobs.

Simple scheduler instruction examples

Here are some commonly used scheduler instructions, along with some examples of their usage:

Setting output file location

To set the output file location for your job, use the `-o <filename>` option - both standard-out and standard-error from your job-script, including any output generated by applications launched by your script, will be saved in the filename you specify.

By default, the scheduler stores data relative to your home-directory - but to avoid confusion, we recommend **specifying a full path to the filename** to be used. Although Linux can support several jobs writing to the same output file, the result is likely to be garbled - it's common practice to include something unique about the job (e.g. it's job-ID) in the output filename to make sure your job's output is clear and easy to read.

Note: The directory used to store your job output file must exist **before** you submit your job to the scheduler. Your job may fail to run if the scheduler cannot create the output file in the directory requested.

For example; the following job-script includes a `-o` instruction to set the output file location:

```
#!/bin/bash -l
#$ -N mytestjob -o $HOME/outputs/output.$JOB_NAME.$JOB_ID

echo "Starting running on host $HOSTNAME"
sleep 120
echo "Finished running - goodbye from $HOSTNAME"
```

In the above example, assuming the job was submitted as user `alces` and was given job-ID number `24`, the scheduler will save output data from the job in the filename `/home/alces/outputs/output.mytestjob.24`.

Setting working directory for your job

By default, jobs are executed from your home-directory on the cluster (i.e. `/home/<your-user-name>`, `$HOME` or `~`). You can include `cd` commands in your job-script to change to different directories; alternatively, you can provide an instruction to the scheduler to change to a different directory to run your job. The available options are:

- `-wd <directory>` - instruct the job scheduler to move into the directory specified before starting to run the job on a compute node
- `-cwd` - instruct the scheduler to move into the same directory you submitted the job from before starting to run the job on a compute node

Note: The directory specified must exist and be accessible by the compute node in order for the job you submitted to run.

Waiting for a previous job before running

You can instruct the scheduler to wait for an existing job to finish before starting to run the job you are submitting with the `-hold_jid <job-ID` instruction. This allows you to build up multi-stage jobs by ensuring jobs are executed sequentially, even if enough resources are available to run them in parallel. For example, to submit a new job that will only start running once job number 15352 has completed, use the following command:

```
qsub -hold_jid 15352 myjobscript.sh
```

Running task array jobs

A common workload is having a large number of jobs to run which basically do the same thing, aside perhaps from having different input data. You could generate a job-script for each of them and submit it, but that's not very convenient - especially if you have many hundreds or thousands of tasks to complete. Such jobs are known as **task arrays** - an embarrassingly parallel job will often fit into this category.

A convenient way to run such jobs on a cluster is to use a task array, using the `-t <start>-<end>[:<step>]` instruction to the job-scheduler. Your job-script can then use pseudo environment variables created by the scheduler to refer to data used by each task in the job. If the `:step` value is omitted, a step value of one will be used. For example, the following job-script uses the `$SGE_TASK_ID` variable to set the input data used for the `bowtie2` application:

```
[alces@login1(scooby) ~]$ cat simplejobscript.sh
#!/bin/bash -l
#$ -N arrayjob
#$ -o $HOME/data/outputs/output.$JOB_ID.$TASK_ID
#$ -t 1-10:2

module load apps/bowtie
bowtie -i $HOME/data/genome342/inputdeck_split.$SGE_TASK_ID -o $HOME/data/outputs/
↪g342.output.$SGE_TASK_ID
```

Note: The pseudo variable `$SGE_TASK_ID` is accessible under the name `$TASK_ID` at submission time (e.g. when setting output file location)

All tasks in a job are given the same job-ID, with the task number indicated after a `.`; e.g.

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
Your job-array 27.1-10:2 ("arrayjob") has been submitted

[alces@login1(scooby) ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          ␣
↪           slots ja-task-ID
-----------------------------------------------------------------------------------
↪---------------------------
     27 11.0273 arrayjob     alces         r     05/15/2016 11:24:29 byslot.q@ip-10-
↪75-0-211.eu-wes     1 1
     27 6.02734 arrayjob     alces         r     05/15/2016 11:24:29 byslot.q@ip-10-
↪75-0-227.eu-wes     1 3
     27 4.36068 arrayjob     alces         r     05/15/2016 11:24:29 byslot.q@ip-10-
↪75-0-201.eu-wes     1 5
     27 3.52734 arrayjob     alces         r     05/15/2016 11:24:29 byslot.q@ip-10-
↪75-0-178.eu-wes     1 7
     27 3.02734 arrayjob     alces         r     05/15/2016 11:24:29 byslot.q@ip-10-
↪75-0-42.eu-west     1 9
```

Individual tasks may be deleted by referring to them using `<job-ID>.<task-ID>` - e.g. to delete task 7 in the above example, you could use the command `qdel 27.7`. Deleting the job-ID itself will delete all tasks in the job.

Requesting more resources

By default, jobs are constrained to the default set of resources (see above) - users can use scheduler instructions to request more resources for their jobs. The following documentation shows how these requests can be made.

Running multi-threaded jobs

If users want to use multiple cores on a compute node to run a multi-threaded application, they need to inform the scheduler - this allows jobs to be efficiently spread over compute nodes to get the best possible performance. Using multiple CPU cores is achieved by requesting access to a **Parallel Environment (PE)** - the default multi-threaded PE on your Alces Flight Compute cluster is called **smp** (for symmetric multi-processing). Users wanting to use the **smp PE** must request it with a job-scheduler instruction, along with the number of CPU cores they want to use - in the form `-pe smp <number-of-cores>`.

For example, to use 4 CPU cores on a single node for an application, the instruction `-pe smp 4` can be used. The following example shows the **smptest** binary being run on 8 CPU cores - this application uses the OpenMP API to automatically detect the number of cores it has available, and prints a simple "hello world" message from each CPU core:

```
[alces@login1(scooby) ~]$ more runsmp.sh
#!/bin/bash -l
#$ -pe smp 8 -o $HOME/smptest/results/smptest.out.$JOB_ID
~/smptest/hello

[alces@login1(scooby) ~]$ qsub runsmp.sh
Your job 30 ("runsmp") has been submitted

[alces@login1(scooby) ~]$ more ~/smptest/results/smptest.out.30
2: Hello World!
5: Hello World!
6: Hello World!
1: Hello World!
4: Hello World!
0: Hello World!
3: Hello World!
7: Hello World!
```

Note: For debugging purposes, an `smp-verbose` PE is also provided that prints additional information in your job output file.

For the best experience, please follow these guidelines when running multi-threaded jobs:

- Alces Flight Compute automatically configures compute nodes with one CPU **slot** per detected CPU core, including hyper-threaded cores.
- **Memory limits** are enforced per CPU-core-slot; for example, if your default memory request is 1.5GB and you request `-pe smp 4`, your 4-core job will be allocated 4 x 1.5GB = **6GB of RAM** in total.
- **Runtime** limits are a measurement of wall-clock time and are not effected by requesting multiple CPU cores.
- Multi-threaded jobs can be **interactive, batch** or **array** type.
- If you request more CPU cores than your largest node can accommodate, your scheduler will print a warning at submission time but still allow the job to queue (in case a larger node is added to your cluster at a later date). For example:

```
[alces@login1(scooby) ~]$ qsub -pe smp 150 simplejobscript.sh
warning: no suitable queues
Your job 58 ("smpjob") has been submitted
```

Note: Requesting more CPU cores in the `smp` parallel environment than your nodes actually have may cause your job to queue indefinitely.

Running Parallel (MPI) jobs

If users want to use run parallel jobs via an install message passing interface (MPI), they need to inform the scheduler - this allows jobs to be efficiently spread over compute nodes to get the best possible performance. Using multiple CPU cores across multiple nodes is achieved by requesting access to a **Parallel Environment (PE)** - the default MPI PE on your Alces Flight Compute cluster is called **mpislots**. Users wanting to use the **mpislots PE** must request it with a job-scheduler instruction, along with the number of CPU cores they want to use - in the form `-pe mpislots <number-of-cores>`. This parallel environment is configured to automatically generate an MPI hostfile, and pass it to the MPI using a scheduler integration.

There is a second parallel environment available which allows users to request complete nodes to participate in MPI jobs - the **mpinodes PE** allows a number of complete nodes to be booked out for jobs that use complete compute nodes at once. Users wanting to use the **mpinodes** PE must request it with a job-scheduler instruction, along with the number of complete nodes they want to use - in the form `-pe mpinodes <number-of-nodes>`.

For example, to use 64 CPU cores on the cluster for a single application, the instruction `-pe mpislots 64` can be used. The following example shows launching the **Intel Message-passing** MPI benchmark across 64 cores on your cluster. This application is launched via the OpenMPI **mpirun** command - the number of threads and list of hosts to use are automatically assembled by the scheduler and passed to the MPI at runtime. This jobscript loads the **apps/imb** module before launching the application, which automatically loads the module for **OpenMPI**.

```
[alces@login1(scooby) ~]$ more runparallel.sh
#!/bin/bash -l
#$ -N IMBjob -pe mpislots 64 -o $HOME/imbjob.out.$JOB_ID

module load apps/imb
mpirun IMB-MPI1

[alces@login1(scooby) ~]$ qsub runparallel.sh
Your job 31 ("IMBjob") has been submitted

[alces@login1(scooby) ~]$ more ~/imbjob.out.31
#------------------------------------------------------------
#    Intel (R) MPI Benchmarks 4.0, MPI-1 part
#------------------------------------------------------------
# Date                    : Mon May 16 12:54:13 2016
# Machine                 : x86_64
# System                  : Linux
# Release                 : 3.10.0-327.18.2.el7.x86_64
# Version                 : #1 SMP Thu May 12 11:03:55 UTC 2016
# MPI Version             : 3.0
# MPI Thread Environment:

# List of Benchmarks to run:
# PingPong, PingPing, Sendrecv, Exchange, Allreduce, Reduce, Reduce_scatter,␣
↪Allgather,
# Allgatherv, Gather, Gatherv, Scatter, Scatterv, Alltoall, Alltoallv, Bcast, Barrier

#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
# ( 62 additional processes waiting in MPI_Barrier)
#---------------------------------------------------
       #bytes #repetitions      t[usec]   Mbytes/sec
```

```
            0         1000         7.25         0.00
            1         1000         7.27         0.13
            2         1000         7.29         0.26
            4         1000         7.32         0.52
            8         1000         7.22         1.06
           16         1000         7.40         2.06
...
```

Note: For debugging purposes, an `mpislots-verbose` PE is also provided that prints additional information in your job output file.

For the best experience, please follow these guidelines when running parallel MPI jobs:

- Alces Flight Compute automatically configures compute nodes with one CPU **slot** per detected CPU core, including hyper-threaded cores.
- **Memory limits** are enforced per CPU-core-slot; for example, if your default memory request is 1.5GB and you request `-pe mpislots 64`, your 64-core job will be allocated `64 x 1.5GB = 96GB` of RAM in total, which may be spread over multiple nodes.
- **Runtime** limits are a measurement of wall-clock time and are not effected by requesting multiple CPU cores.
- Parallel jobs can be interactive, batch or array type.
- Parallel applications must use an MPI to handle multi-node communications; the scheduler will prepare nodes for use, but users must use an MPI to launch the application (as shown in the example above).
- If you request more CPU cores than your cluster can accommodate, your scheduler will print a warning at submission time but still allow the job to queue (in case more nodes are added to your cluster at a later date). For example:

```
[alces@login1(scooby) ~]$ qsub -pe mpislots 1024 runparallel.sh
warning: no suitable queues
Your job 32 ("IMBjob") has been submitted
```

Note: Requesting more CPU cores in the `mpislots` parallel environment than your nodes actually have may cause your job to queue indefinitely. If auto-scaling is enabled, the cluster will be expanded over a few minutes to its maximum size in an attempt to add resources to allow your job to run. If you request more resources than your auto-scaling limit will allow, your job may queue indefinitely.

Requesting more memory

Your jobs are restricted to using a maximum amount of memory on the compute node they are executed on. The default memory allocation divides the total amount of RAM per node by the number of available CPU cores - e.g. a cluster that has node with 36 cores and 160GB of RAM will have a default memory allocation of 4.4GB. This allows the job scheduler to efficiently manage resources, ensuring all jobs get enough memory to run without the node running out of memory and crashing.

If you need more than the default amount of memory for your job, use the `-l h_vmem=<amount-of-RAM>` scheduler instruction to request more. For example, to request 32GB of RAM for your single-CPU interactive job, you can use the command `qrsh -l h_vmem=32G`:

```
[alces@login1(scooby) ~]$ qrsh -l h_vmem=32G

<<< -[ alces flight ]- >>>
[alces@ip-10-75-0-128(scooby) ~]$
```

Memory allocations are performed **per scheduler slot** - i.e. per CPU core. So - if you want to request to run an 8-CPU-core multi-threaded job with a total of 64GB of memory, you would request `-pe smp 8 -l h_vmem=8G` (as 64GB / 8-cores = **8GB per core**).

Note: Memory allocations are automatically enforced by the job scheduler. If your application exceeds it's memory request, your job will be stopped to prevent crashing the hosting compute node.

How do I know how much memory to request?

It can be difficult for new users to know how much memory their job needs to run effectively. Use the method below to run your job with a high memory limit in order to identify appropriate memory limits to request:

1. Use the `qhost` command to view how much memory your nodes have (**MEMTOT** column).
2. Submit your job with the maximum memory request for your node type.

   e.g. If your nodes are reported as having 58GB of RAM:

   - Submit a single-CPU job with the instruction `-l h_vmem=58G`
   - Submit a 4-CPU core multi-threaded job with the instruction `-l h_vmem=14.5G`
   - Submit an 8-CPU core multi-threaded job with the instruction `-l h_vmem=7.25G`

   etc.
3. Note the job-ID number of your job while it is running, and allow your job to finish normally.
4. Use the `qacct -j <job-ID>` command to view the scheduler accounting database entry for your job
5. Look for the entry in the `qacct` output that starts **maxvmem** - this will display how much memory your job used when running

The next time you submit your job, you can use a smaller memory request for your job, based on the information you gathered from the `qacct` output.

Note: A number of parameters can effect how much memory a job uses when running, including the node type and data-set size. Most users round-up their memory requests to the nearest whole GB of RAM.

Requesting a longer runtime

By default, the scheduler imposes a maximum runtime of 24-hours for jobs submitted on your cluster. This ensures that run-away jobs do not continue processing for long periods of time without generating useful output. Users can request a longer runtime (with no upper limit) by using the `-l h_rt=<hours>:<mins>:<secs>` scheduler instruction. For example, to submit a job-script called `longjob.sh` with a 72-hour runtime, use the following command:

```
[alces@login1(scooby) ~]$ qsub -l h_rt=72:0:0 longjob.sh
Your job 39 ("longjob.sh") has been submitted
```

Job-script templates

Your Alces Flight Compute cluster includes a number of job-script templates for common types of non-interactive jobs. The templates come complete with a range of different scheduler instructions built-in and are designed to use as the basis of your own job-scripts.

To view the available template for your Flight Compute cluster, use the `alces template list` command:

```
[alces@login1(scooby) ~]$ alces template list
 1 -> mpi-nodes    ... MPI multiple node
 2 -> mpi-slots    ... MPI multiple slot
 3 -> simple-array ... Simple serial array
 4 -> simple       ... Simple serial
 5 -> smp          ... SMP multiple slot
```

Templates can be previewed using their number or description - e.g. to look at the template for an array job based on the output above, use the command `alces template show simple-array`.

To create a new job-script based on a template, use the `alces template copy <template-name> <job-script-filename>` command; e.g.

```
[alces@login1(scooby) ~]$ alces template copy smp mysmpjob.sh
alces template copy: template 'smp' copied to 'mysmpjob.sh'

[alces@login1(scooby) ~]$ nano mysmpjob.sh
   <edit template to add include details of my application>

[alces@login1(scooby) ~]$ qsub mysmpjob.sh
Your job 41 ("mysmpjob.sh") has been submitted

[alces@login1(scooby) ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          ␣
↪            slots ja-task-ID
-----------------------------------------------------------------------------------------
↪-----------------------------
     41 2.40234 mysmpjob.s alces        r     05/15/2016 13:39:34 byslot.q@ip-10-75-0-
↪114.eu-wes     2
```

Trouble-shooting

Your cluster job-scheduler is capable of running complex workflows, utilising advanced features to control every facet of running your jobs. It's worth reading through the job-script template to look at the common option available, and trying out different options before running production jobs on your cluster.

If you do run into problems, the `qstat -j <job-id>` command can be useful - as well as showing you the instructions you passed the scheduler with your job, the output of this command will also show you the current environment settings for your job, and list scheduling information. This can provide you with assistance to debug issues, and explain why jobs are still queuing when you think they should be running.

Be patient with the job-scheduler if you have auto-scaling enabled - queuing jobs cannot start until new compute nodes have succesfully joined the cluster; the speed of scaling-up the cluster is governed by the performance of your Cloud provider, and the amount you've paid for your instance types.

Further documentation

We have just scratched the surface of using a cluster job-scheduler, and there are many more features and options than described here. A wide range of documentation available both on your Flight Compute cluster and online;

- Use the `man qsub` command for a full list of scheduler instructions
- Use the `man qstat` command to see the available reporting options for running jobs
- Use the `man qacct` command to see options for accessing the job-accounting database
- Online documentation for the Grid-Engine scheduler series is available here

Customising your job-scheduler

Your Alces Flight Compute cluster has been pre-configured with default queues, parallel-environments and resource limits based on a known working set used internationally by HPC sites and research organisations. They have been determined to deliver a good balance of safety and flexibility, and are designed to introduce concepts such as requesting resources which are commonplace for many HPC facilities.

However - your personal Flight Compute cluster can be modified to suit whatever you need it to do. There are no right and no wrong answers here - you have full control over your facility to do whatever you want. Your login user is authorized to make configuration changes to the scheduler as desired - you can also use the `sudo` command to become the root-user to make any other changes you require.

There is a graphical administration interface for the OGS scheduler - to use it, follow these instructions:

1. Install the **Motif** software package and fonts needed by the OGS GUI; use the command `sudo yum install motif xorg-x11-fonts-*`
2. Use the `alces session start gnome` command to start a graphical desktop session, if you don't already have one.
3. Start the graphical interface using the command `qmon`

Be aware that any job-scripts you have already created (including the provided templates) and cluster auto-scaling support (if available) may rely to the default configuration delivered with your Flight Compute cluster. If you re-configure the scheduler, we recommend that you disable auto-scaling (`alces configuration autoscaling disable`) and review your job-scripts for compatibility.

Cluster customisation

An Alces Flight Compute environment contains many useful configurations and tools, however there are cases where your environment may require customisation - this may include things such as;

- Distribution package installation
- External storage mounts

The Alces customiser tool requires some setup tasks in order to appropriately work with your AWS account and deployed Alces Flight Compute environments.

Setup tasks

To begin setting up the Alces customiser tool - log in to your Alces Flight Compute environment as the administrator user. From there, run the `alces about node` command - this will display information about your environment:

```
[alces@login1(alces-cluster) ~]$ alces about node
Clusterware release: 2016.2
Customizer bucket prefix: s3://alces-flight-a1i0ytdmvzv3ztv3/customizer/default
Platform host name: ec2-52-51-77-141.eu-west-1.compute.amazonaws.com
Public IP address: 52.51.77.141
Account hash: a1i0ytdmvzv3ztv3
```

Using the information provided - a new S3 bucket must be created using the prefix provided. Using one of the available tools, such as `alces storage` or `s3cmd` or the S3 web console - create a new bucket, for instance in the example the bucket would be called:

    alces-flight-a1i0ytdmvzv3ztv3

Once the bucket is created - create a "folder" within the bucket named:

    customizer

Now, from within the `customizer` folder - create another folder named `default`, this sets up the `default` customiser profile.

From within the `default` folder - create the configuration folder - this is where all customisation scripts should be placed. Create a new folder within the `default` folder named:

    configure.d

Note: It is important that the bucket and folders are created with the correct names - failing to create the bucket and folders with the correct names will mean the Alces customiser will not run

Your AWS account is now ready for use with the Alces customiser tool.

Setting up customisation scripts

Customisation scripts are run on each node in your environment upon entering the cluster ring - example customisation scripts include distribution package installations and external storage mounts. The Alces customiser supports any executable file type.

The following simple example customisation shell script would install the `emacs` editor on each node within your environment:

```
#!/bin/bash
yum -y install emacs
```

Once the bash script has been created - upload it to your S3 bucket into the `configure.d` folder previously created, for example:

    s3://alces-flight-<account hash>/customizer/default/configure.d/
    emacs.sh

You can upload multiple customisation scripts to the `default` folder - each of the scripts will be run.

The output of each customiser script run is sent to `/var/log/clusterware/instance.log` on each of the nodes. Each subsequently deployed Alces Flight Compute environment will run each of the customise scripts included in the `default` folder.

Working with Gridware Applications: R

The following guide will detail the installation of R, along with some of the advanced features of Alces Gridware when working with the R Gridware package.

R Installation

Many different versions of R are available through the Alces Gridware utility - you can see the list of available packages with the `alces gridware list` function, for example:

```
[alces@login1(hpc1) ~]$ alces gridware list R
base/apps/R/2.15.0  base/apps/R/2.15.1  base/apps/R/2.15.2  base/apps/R/2.15.3
base/apps/R/3.0.0   base/apps/R/3.0.1   base/apps/R/3.1.1   base/apps/R/3.1.2
base/apps/R/3.2.0   base/apps/R/3.2.1   base/apps/R/3.2.2
```

To install, for example - R version 3.2.2; run the following command:

```
[alces@login1(hpc1) ~]$ alces gridware install R/3.2.2
Installing base/apps/R/3.2.2
ERROR: Unable to satisfy compilation requirements: libs/lapack, libs/blas
```

The Alces Gridware tool automatically checks dependencies for you - in this case, we do not have some of the required libraries needed to compile the R application. First - install the `libs/lapack` and `libs/blas` Gridware packages, you will then be able to proceed with the `R 3.2.2` installation:

```
[alces@login1(hpc1) ~]$ alces gridware install R/3.2.2
Installing base/apps/R/3.2.2
 > Preparing package sources
        Download --> R-3.2.2.tar.gz ... SKIP (Existing source file detected)
          Verify --> R-3.2.2.tar.gz ... OK
        Packaged --> Rprofile       ... OK

 > Preparing for installation
           Mkdir ... OK (/var/cache/gridware/src/apps/R/3.2.2/gcc-4.8.5+lapack-3.5.
↪0+blas-20110419)
         Extract ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK
(/opt/gridware/depots/2b8a9f1c/el7/pkg/apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-
↪20110419)
         Install ... OK
          Module ... OK

Installation complete.
```

Once the compilation has finished - the R 3.2.2 Gridware package will be available for use, check its availability and load using:

```
[alces@login1(hpc1) ~]$ module avail
---  /opt/gridware/local/el7/etc/modules  ---
  apps/perl/5.18.0/gcc-4.8.5
  apps/perl/5.20.2/gcc-4.8.5
  apps/python/2.7.5/gcc-4.8.5
  apps/python/2.7.8/gcc-4.8.5
  apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
  compilers/gcc/system
  libs/blas/20110419/gcc-4.8.5
  libs/gcc/system
  libs/lapack/3.5.0/gcc-4.8.5
  null
---  /opt/clusterware/etc/modules  ---
  null
  services/gridscheduler
[alces@login1(hpc1) ~]$ module load apps/R
apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
 | -- libs/gcc/system ... SKIPPED (already loaded)
 |
 OK
[alces@login1(hpc1) ~]$ R --version
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
```

Multiple versions of a package can exist at one time, however only one version of a particular application can be loaded at any one time - to load a different version of R:

```
[alces@login1(hpc1) ~]$ alces module load apps/R/3.1.2
apps/R/3.1.2/gcc-4.8.5+lapack-3.5.0+blas-20110419 ... VARIANT (have alternative: apps/
↪R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419)
[alces@login1(hpc1) ~]$ alces module unload apps/R/3.2.2
apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419 ...
                                                   UNLOADING --> OK
[alces@login1(hpc1) ~]$ alces module load apps/R/3.1.2
apps/R/3.1.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
 | -- libs/gcc/system ... SKIPPED (already loaded)
 |
 OK
[alces@login1(hpc1) ~]$ R --version
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
```

Installation of language libraries

Through the Alces Gridware utility, installation of lanaguage libraries is possible both on a system-wide level, and also on a per-user basis. The following section details both system-wide language library installation, as well as user-level language library installation.

System-wide language libraries: R

As the `alces` administrator user, or any other sudo enabled user that can switch to root - change to the `root` user account.

To add R packages, first load the version of R you wish to install packages to - for example `apps/R/3.2.2`:

```
[root@login1(hpc1) ~]# module load apps/R/3.2.2
apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
 | -- libs/gcc/system
 |     * --> OK
 |
 OK
```

Next, load the `R` application - and use the `install.packages` command to install your desired system-wide packages:

```
[root@login1(hpc1) ~]# R
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

> install.packages("randomForest", repos="http://cran.cnr.berkeley.edu")
Installing package into ‘/opt/gridware/share/R/3.2.2’
(as ‘lib’ is unspecified)
trying URL 'http://cran.cnr.berkeley.edu/src/contrib/randomForest_4.6-12.tar.gz'
<-- snip -->
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
```

Once the installation is complete and you have verified the package works as intended, yo ucan check the package is available to other users on the system:

```
[barney@login1(hpc1) ~]$ module load apps/R/3.2.2
apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
 | -- libs/gcc/system
 |    * --> OK
 |
 OK
[barney@login1(hpc1) ~]$ R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
<-- snip -->
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
```

User-specific language libraries: R

Users may also wish to install their own language libraries, these will be unavailable to other users of the environment.

As the user you wish to install an R package for, load the version of R you wish to install the packages for (e.g. `apps/R/3.2.2`).

After the R application is loaded, use the `install.packages("packagename")` function to install packages you require - for example:

```
[barney@login1(hpc1) ~]$ module load apps/R/3.2.2
apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
 | -- libs/gcc/system ... SKIPPED (already loaded)
 |
 OK
[barney@login1(hpc1) ~]$ R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
<-- snip -->
```

```
> install.packages("snow")
Installing package into '/home/barney/gridware/share/R/3.2.2'
(as 'lib' is unspecified)
<-- snip -->
> library(snow)
> packageVersion("snow")
[1] '0.4.1'
```

The `snow` package installation was successful - and we can now use it as the `barney` user. Switching to another user will confirm the user-level installation success, the `alces` user will not be able to use the `snow` R package:

```
[alces@login1(hpc1) ~]$ module load apps/R/3.2.2
apps/R/3.2.2/gcc-4.8.5+lapack-3.5.0+blas-20110419
 | -- libs/gcc/system
 |    * --> OK
 |
 OK
[alces@login1(hpc1) ~]$ R

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
<-- snip -->
> library(snow)
Error in library(snow) : there is no package called 'snow'
```

Working with Gridware Applications: Perl

The following guide will detail the installation of Perl, along with some of the advanced features of Alces Gridware when working with the Perl Gridware package.

Perl Installation

Many different versions of Perl are available through the Alces Gridware utility - you can see the list of available packages with the `alces gridware search` function, for example:

```
[alces@login1(hpc1) ~]$ alces gridware search --name perl
base/apps/perl/5.10.1          base/apps/perl/5.12.4
base/apps/perl/5.14.2          base/apps/perl/5.16.1
base/apps/perl/5.16.3          base/apps/perl/5.18.0
base/apps/perl/5.20.2          base/apps/perl/5.8.8
base/apps/perl/5.8.9           base/apps/sierraperl/20080201
base/libs/bioperl/1.2.3        base/libs/bioperl/1.6.901
base/libs/bioperl/1.6.923
```

To install, for example - Perl version 5.20.2; run the following command:

```
[alces@login1(hpc1) ~]$ alces gridware install perl/5.20.2
Installing base/apps/perl/5.20.2

 > 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 (/var/cache/gridware/src/apps/perl/5.20.2/gcc-4.8.5)
         Extract ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK (/opt/gridware/depots/2b8a9f1c/el7/pkg/apps/perl/5.20.2/gcc-4.
↪8.5)
         Install ... OK
          Module ... OK

Installation complete.
```

Once the compilation has finished - the Perl 5.20.2 Gridware package will be available for use, check its availability and load using:

```
[alces@login1(hpc1) ~]$ module avail
---  /opt/gridware/local/el7/etc/modules  ---
  apps/perl/5.20.2/gcc-4.8.5
[alces@login1(hpc1) ~]$ module load apps/perl
apps/perl/5.20.2/gcc-4.8.5
 | -- libs/gcc/system ... SKIPPED (already loaded)
 |
 OK
[alces@login1(hpc1) ~]$ perl --version
This is perl 5, version 20, subversion 2 (v5.20.2) built for x86_64-linux versions of␣
↪Python can be installed at once using Gridware and Modules - for example:
```

Multiple versions of a package can exist at one time, however only one version of a particular application can be loaded at any one time - to load a different version of Perl:

```
[alces@login1(hpc1) ~]$ alces module load apps/perl/5.18.0/gcc-4.8.5
apps/perl/5.18.0/gcc-4.8.5 ... VARIANT (have alternative: apps/perl/5.20.2/gcc-4.8.5)
[alces@login1(hpc1) ~]$ alces module unload apps/perl/5.20.2/gcc-4.8.5
              apps/perl/5.20.2/gcc-4.8.5 ... UNLOADING --> OK
[alces@login1(hpc1) ~]$ alces module load apps/perl/5.18.0/gcc-4.8.5
apps/perl/5.18.0/gcc-4.8.5
 | -- libs/gcc/system ... SKIPPED (already loaded)
 |
 OK
[alces@login1(hpc1) ~]$ perl --version
This is perl 5, version 18, subversion 0 (v5.18.0) built for x86_64-linux
```

Installation of language libraries

Through the Alces Gridware utility, installation of lanaguage libraries is possible both on a system-wide level, and also on a per-user basis. The following section details both system-wide language library installation, as well as user-level language library installation.

System-wide language libraries: Perl

As the `alces` administrator user, or any other sudo enabled user that can switch to root - change to the `root` user account.

Next, load the version of Perl you wish to add language libraries to - for example `perl/5.20.2`

```
[root@login1(hpc1) ~]# module load apps/perl/5.20.2
apps/perl/5.20.2/gcc-4.8.5
 | -- libs/gcc/system
 |     * --> OK
 |
 OK
```

Next - use the `cpan` utility to install the Perl libraries you, or additional system users require - for example:

```
[root@login1(hpc1) ~]# cpan Date::Simple
Fetching with Net::FTP:
ftp://cpan.etla.org/pub/CPAN/authors/01mailrc.txt.gz
Reading '/opt/gridware/share/perl/5.20.2/cpan/sources/authors/01mailrc.txt.gz'
<--snip-->
```

The `Date::Simple` module will now be available to any system user loading the `Perl 5.20.2` Gridware package.

To verify successful installation, switch to a non-root user; for example `barney` will now be able to see and use the `Date::Simple` module:

```
[barney@login1(hpc1) ~]$ module load apps/perl/5.20.2
apps/perl/5.20.2/gcc-4.8.5
 | -- libs/gcc/system
 |     * --> OK
 |
 OK
[barney@login1(hpc1) ~]$ cpan -l 2>&1 | grep Date::Simple | head -n1
Date::Simple        3.03
```

User-specific language libraries: Perl

Users may also wish to install their own language libraries, these will be unavailable to other users of the environment.

As the user you wish to install a Perl module for, load the `perl` Gridware application, then use `cpan` to install the required module:

```
[barney@login1(hpc1) ~]$ cpan File::Slurp
Fetching with Net::FTP:
ftp://cpan.etla.org/pub/CPAN/authors/01mailrc.txt.gz
Reading '/home/barney/gridware/share/perl/5.20.2/cpan/sources/authors/01mailrc.txt.gz'
<-- snip -->
[barney@login1(hpc1) ~]$ cpan File::Slurp
Reading '/home/barney/gridware/share/perl/5.20.2/cpan/Metadata'
  Database was generated on Fri, 19 Feb 2016 02:41:02 GMT
File::Slurp is up to date (9999.19).
```

The `File::Slurp` installation was successful - and we can now use it as the `barney` user. Switching to another user will confirm the user-level installation success, the `alces` user will not be able to use the `File::Slurp` Perl module, and instead try to make them install the `File::Slurp` module:

```
[alces@login1(hpc1) ~]$ alces module load apps/perl/5.20.2
[alces@login1(hpc1) ~]$ cpan File::Slurp
Fetching with Net::FTP:
```

```
ftp://cpan.etla.org/pub/CPAN/authors/01mailrc.txt.gz
Reading '/home/alces/gridware/share/perl/5.20.2/cpan/sources/authors/01mailrc.txt.gz'
```

Working with Gridware Applications: Python

The following guide will detail the installation of Python, along with some of the advanced features of Alces Gridware when working with the Python Gridware package.

Python Installation

Many different versions of Python are available through the Alces Gridware utility - you can see the list of available packages with the `alces gridware search` function, for example:

```
[alces@login1(hpc1) ~]$ alces gridware search --name python
base/apps/ipython/2.3.0   base/apps/python/2.7.3    base/apps/python/2.7.5
base/apps/python/2.7.8    base/apps/python3/3.2.3   base/apps/python3/3.3.3
base/apps/python3/3.4.0   base/apps/python3/3.4.3   base/libs/biopython/1.61
base/libs/biopython/1.63
```

To install, for example - Python version 2.7.8; run the following command:

```
    [alces@login1(hpc1) ~]$ alces gridware install python/2.7.8
    Installing base/apps/python/2.7.8

 > Preparing package sources
        Download --> Python-2.7.8.tgz ... SKIP (Existing source file detected)
      Verify --> Python-2.7.8.tgz ... OK

 > Preparing for installation
           Mkdir ... OK (/var/cache/gridware/src/apps/python/2.7.8/gcc-4.8.5)
         Extract ... OK
    Dependencies ... OK

 > Proceeding with installation
         Compile ... OK
           Mkdir ... OK (/opt/gridware/depots/2b8a9f1c/el7/pkg/apps/python/2.7.8/gcc-
↪4.8.5)
         Install ... OK
          Module ... OK

Installation complete.
```

Once the compilation has finished - the Python 2.7.8 Gridware package will be available for use, check its availability and load using:

```
[alces@login1(hpc1) ~]$ alces module load apps/python
apps/python/2.7.8/gcc-4.8.5
 | -- libs/gcc/system
 |    * --> OK
 |
 OK
[alces@login1(hpc1) ~]$ python --version
Python 2.7.8
```

Multiple versions of Python can be installed at once using Gridware and Modules - for example:

```
[alces@login1(hpc1) ~]$ alces module avail
---  /opt/gridware/local/el7/etc/modules  ---
  apps/python/2.7.5/gcc-4.8.5
  apps/python/2.7.8/gcc-4.8.5
```

Only one version of a particular application can be loaded at any one time - to load a different version of Python:

```
[alces@login1(hpc1) ~]$ alces module load apps/python/2.7.5/gcc-4.8.5
apps/python/2.7.5/gcc-4.8.5 ... VARIANT (have alternative: apps/python/2.7.8/gcc-4.8.
↪5)
[alces@login1(hpc1) ~]$ alces module unload apps/python/2.7.8/gcc-4.8.5
             apps/python/2.7.8/gcc-4.8.5 ... UNLOADING --> OK
[alces@login1(hpc1) ~]$ alces module load apps/python/2.7.5/gcc-4.8.5
apps/python/2.7.5/gcc-4.8.5
 | -- libs/gcc/system ... SKIPPED (already loaded)
 |
 OK
[alces@login1(hpc1) ~]$ python --version
Python 2.7.5
```

Installation of language libraries

Through the Alces Gridware utility, installation of lanaguage libraries is possible both on a system-wide level, and also on a per-user basis. The following section details both system-wide language library installation, as well as user-level language library installation.

System-wide language libraries: Python

As the `alces` administrator user, or any other sudo enabled user that can switch to root - change to the `root` user account.

To add Python packages, the `setuptools` Gridware application is required - this can be installed using `alces gridware install setuptools/15.1 --variant default`. Once the `setuptools` module is available, load it as the `root` user:

```
[root@login1(hpc1) ~]# module load apps/setuptools
apps/setuptools/15.1/python-2.7.8
 | -- apps/python/2.7.8/gcc-4.8.5
 |     | -- libs/gcc/system
 |     |     * --> OK
 |     * --> OK
 |
 OK
```

Next, using `easy_install` - install the Python libraries required, for example:

```
[root@login1(hpc1) ~]# easy_install numpy
Creating /opt/gridware/share/python/2.7.8/lib/python2.7/site-packages/site.py
Searching for numpy
Reading https://pypi.python.org/simple/numpy/
Best match: numpy 1.11.0b3
<-- snip -->
Installed /opt/gridware/share/python/2.7.8/lib/python2.7/site-packages/numpy-1.11.0b3-
↪py2.7-linux-x86_64.egg
```

```
Processing dependencies for numpy
Finished processing dependencies for numpy
```

Once the installation is complete - you can check the library is available to other users on the system:

```
[barney@login1(hpc1) ~]$ module load apps/python/2.7.8
apps/python/2.7.8/gcc-4.8.5
 | -- libs/gcc/system
 |    * --> OK
 |
 OK
[barney@login1(hpc1) ~]$ python
Python 2.7.8 (default, Feb 19 2016, 10:02:41)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.version.version
'1.11.0b3'
```

User-specific language libraries: Python

Users may also wish to install their own language libraries, these will be unavailable to other users of the environment.

As the user you wish to install a Python library for, load the `setuptools` Gridware application for the version of Python you wish to install libraries for (e.g. `apps/setuptools/15.1/python-2.7.8`), then use `easy_install` to install the required module:

```
[barney@login1(hpc1) ~]$ easy_install htseq
Searching for htseq
Reading https://pypi.python.org/simple/htseq/
Best match: HTSeq 0.6.1
<-- snip -->
Installed /home/barney/gridware/share/python/2.7.8/lib/python2.7/site-packages/HTSeq-
↪0.6.1-py2.7-linux-x86_64.egg
Processing dependencies for htseq
Finished processing dependencies for htseq
[barney@login1(hpc1) ~]$ python
Python 2.7.8 (default, Feb 19 2016, 10:02:41)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import HTSeq
>>> HTSeq.__version__
'0.6.0'
```

The `htseq` installation was successful - and we can now use it as the `barney` user. Switching to another user will confirm the user-level installation success, the `alces` user will not be able to use the `HTSeq` Python library:

```
[alces@login1(hpc1) ~]$ module load apps/python
apps/python/2.7.8/gcc-4.8.5
 | -- libs/gcc/system
 |    * --> OK
 |
 OK
[alces@login1(hpc1) ~]$ python
Python 2.7.8 (default, Feb 19 2016, 10:02:41)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-4)] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import HTSeq
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named HTSeq
```

Using OpenFoam with Alces Flight Compute

The following guide will run through the basics of using OpenFOAM together with an Alces Flight Compute environment.

Prerequisites

- Alces Flight Compute environment deployed with at least 2 compute nodes

Installing the OpenFOAM Gridware Depot

The OpenFOAM Gridware depot can be easily installed, providing you instant access to the OpenFOAM application.

As an authorised user, download and enable the OpenFOAM Gridware Depot:

```
[alces@login1(hpc1) ~]$ alces gridware depot fetch https://s3-eu-west-1.amazonaws.com/
↪packages.alces-software.com/depots/openfoam

 > Fetching depot
        Metadata ... OK
         Content ... OK
         Extract ... OK
            Link ... OK

 > Resolving depot dependencies: openfoam

Depot 'openfoam' fetched successfully.
[alces@login1(hpc1) ~]$ alces gridware depot enable openfoam

 > Enabling depot: openfoam
          Enable ... OK
```

Manual installation of OpenFOAM

Alternatively, you may wish to install the Gridware applications yourself - this can be done by installing the following Gridware packages in order:

```
alces gridware install openmpi/1.8.5 qlogic=false torque=false pmi=false pmilib=false␣
↪sge=true
alces gridware install libs/scotch
alces gridware install libs/mgridgen
alces gridware install openfoam/2.1.1
alces gridware install paraview/4.3.1
```

Running OpenFOAM

The following tutorial makes use of the OpenFOAM graphical interface. To use the graphical interface, a GNOME desktop session should be started. Sessions can easily be created using `alces session`. Create a GNOME desktop session and connect to it using your favourite VNC client:

```
[alces@login1(hpc1) ~]$ alces session start gnome
VNC server started:
    Identity: 36a814b0-dc84-11e5-bcf2-fa163e8729ee
        Type: gnome
        Host: 10.77.2.129
        Port: 5901
     Display: 1
    Password: vvrDZM2Z
   Websocket: 41361

Depending on your client, you can connect to the session using:

  vnc://alces:vvrDZM2Z@10.77.2.129:5901
  10.77.2.129:5901
  10.77.2.129:1

If prompted, you should supply the following password: vvrDZM2Z
```

Loading OpenFOAM

Once you have connected to the VNC session - the OpenFOAM application will need to be loaded.

1. Open the `Terminal` application
2. Load the OpenFOAM module

```
module load apps/openfoam
```

3. Using the Terminal session, navigate to the tutorials directory. The `$FOAM_TUTORIALS` environment variable is automatically set when loading the OpenFOAM module, and will take you to the correct location:

```
[alces@login1(hpc1) ~]$ cd $FOAM_TUTORIALS
[alces@login1(hpc1) tutorials]$ ls
Allclean  basic          discreteMethods   financial       lagrangian  resources
Allrun    combustion     DNS               heatTransfer    mesh        stressAnalysis
Alltest   compressible   electromagnetics  incompressible  multiphase
```

4. Make a copy of the `cavity` tutorial to your home directory

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity $HOME/cavity
```

5. Navigate to the `cavity` directory in your home folder. From here we can create the mesh using the available OpenFOAM tools. From the `cavity` directory, run the `blockMesh` command - this will generate a mesh in OpenFOAM format:

```
[alces@login1(hpc1) cavity]$ blockMesh
Build  : 2.2.1-57f3c3617a2d
Exec   : blockMesh
Date   : Feb 26 2016
Time   : 14:59:24
Host   : "login1"
```

```
PID    : 12720
Case   : /home/alces/cavity
nProcs : 1
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Disallowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Creating block mesh from
    "/home/alces/cavity/constant/polyMesh/blockMeshDict"
Creating curved edges
Creating topology blocks
Creating topology patches

Creating block mesh topology

Check topology

        Basic statistics
                Number of internal faces : 0
                Number of boundary faces : 6
                Number of defined boundary faces : 6
                Number of undefined boundary faces : 0
        Checking patch -> block consistency

Creating block offsets
Creating merge list .

Creating polyMesh from blockMesh
Creating patches
Creating cells
Creating points with scale 0.1

Writing polyMesh
----------------
Mesh Information
----------------
  boundingBox: (0 0 0) (0.1 0.1 0.01)
  nPoints: 882
  nCells: 400
  nFaces: 1640
  nInternalFaces: 760
----------------
Patches
----------------
  patch 0 (start: 760 size: 20) name: movingWall
  patch 1 (start: 780 size: 60) name: fixedWalls
  patch 2 (start: 840 size: 800) name: frontAndBack

End
```

6. You can verify success, and view information such as mesh size, geometrical size and some mesh checks using the `meshCheck` command.
7. You've now created a case for the solver - which we can run using OpenFOAM. To run the process interactively, perform the following command:

```
icoFoam
Build  : 2.2.1-57f3c3617a2d
Exec   : icoFoam
Date   : Feb 26 2016
Time   : 15:04:13
Host   : "login1"
PID    : 13173
Case   : /home/alces/cavity
nProcs : 1
fileModificationChecking : Monitoring run-time modified files using timeStampMaster
allowSystemOperations : Disallowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Create mesh for time = 0
<-- snip -->
```

Alternatively - the process can be automated through your cluster job scheduler.

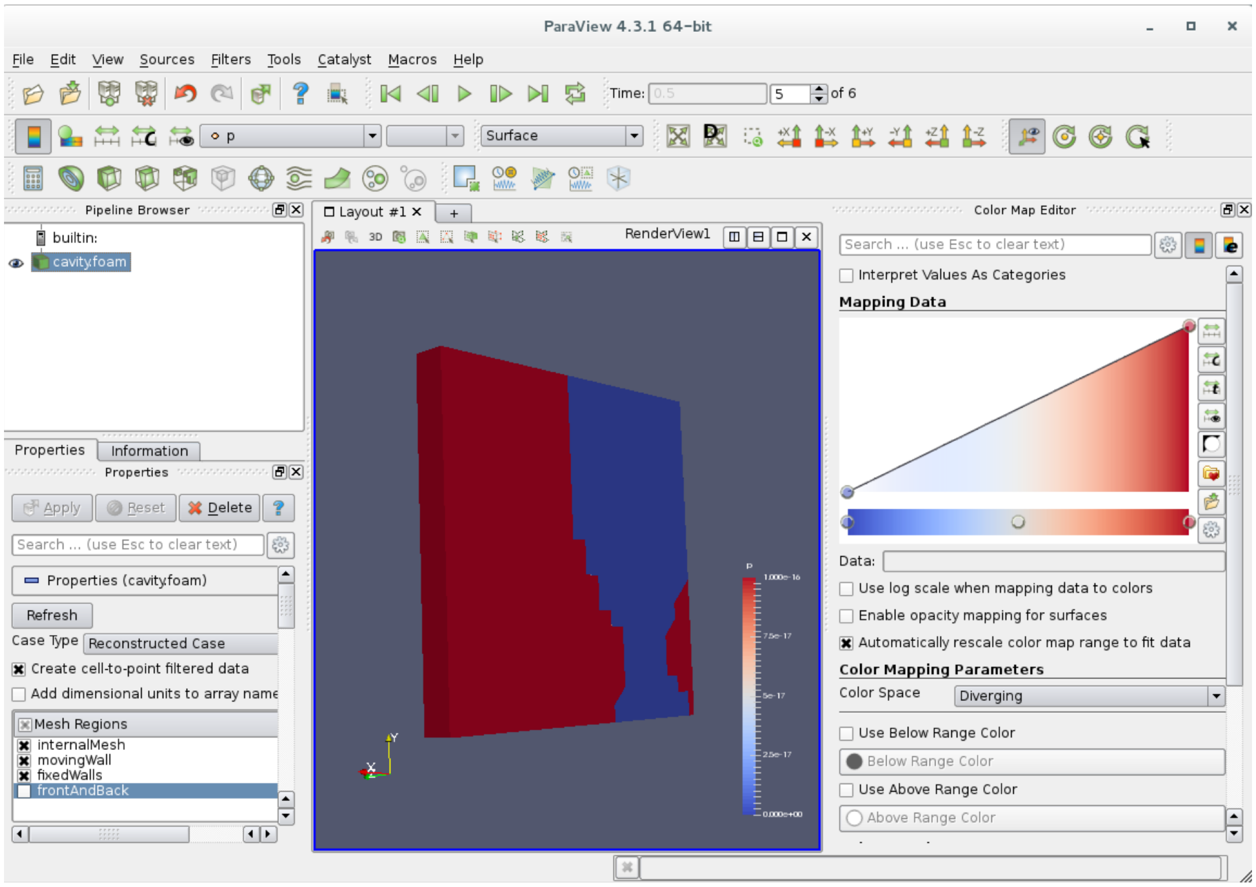
8. Now that you have completed your solve, you may wish to view the post-processing results. From your Terminal session, load the `paraview` application:

```
module load apps/paraview
```

9. From the `cavity` directory in your home folder, run the viewer - this will open up the paraFoam viewer interface:

```
paraFoam -builtin
```

10. Using the `Mesh Regions` box on the bottom left of the interface - enable all of the Mesh regions.
11. Click the `Play` button using the toolbar to run the output.



Case Type: Reconstructed Case

[✓] Create cell-to-point filtered data

[ ] Add dimensional units to array name

Mesh Regions
- [✓] internalMesh
- [✓] movingWall
- [✓] fixedWalls
- [ ] frontAndBack

[ ] Interpret Values As Categories

[ ] Use log scale when mapping data to colors

[ ] Enable opacity mapping for surfaces

[✓] Automatically rescale color map range to fit data

Color Space: Diverging

[ ] Use Below Range Color

[ ] Use Above Range Color