
flight-appliance-docs Documentation

Release 1.0

Alces Flight Ltd

Jul 26, 2017

Using your Flight Compute cluster

1	License	3
2	Prerequisites	5



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

This documentation is released under the [Creative-Commons: Attribution-ShareAlike 4.0 International](#) license. The Alces Flight Compute software is released under the terms of the [Alces Flight EULA](#). Please read the relevant license text before proceeding to use.

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 designed to provide a quick-start guide to your personal Flight Compute cluster, but if you've used clusters before then you might just need a *super-quick* guide to help you get started. 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. To use the AWS platform, use the **Personal HPC compute cluster** launch option from the [Alces Flight Marketplace product page](#) to start a personal cluster in a new VPC. You will be asked various questions about how you want your cluster to look - most options are explained, with sensible defaults shown.
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 one-time password shown 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. The Flight Compute product is designed to support a single user, although multiple users can login using the same account.

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, Swift 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 a batch job scheduler; the default is Open Grid Scheduler (SGE), but a range of other schedulers are also available. 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 Compute is not intended 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 trial 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. The team at Alces package both a community supported version which is free to use, and a commercially supported version which customers can buy to get access to more features and professional assistance. Alces also provide multi-user clusters based on Flight for both on-premise and public-cloud platforms - if you enjoy using Flight Compute, contact Alces for more information on how you can buy your own multi-user environment.

You are likely to incur infrastructure costs from your platform provider (i.e. AWS, or your local OpenStack team) if you are running your cluster 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 - 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 a CloudFormation 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

Method of Launching

The simplest method of launching a cluster is by using the AWS Marketplace - clusters launch using an AWS CloudFormation template which asks the users a number of simple questions in order to configure their cluster environment. This method is documented on this page, and is the fastest way to launch your own, personal HPC cluster environment.

Advanced users may also wish to launch a cluster one instance at a time, or deploy a single login node to be used interactively. Follow this guide for information on how to manually configure a cluster by launching individual instances - [Launching a single Alces Flight instance on AWS](#).

Users can also use one the example CloudFormation templates as the basis of their own cluster deployments. This allows more customisation of your cluster, including choosing how many nodes can be launched, configuring different types of EBS backing storage and choosing different availability zones for your compute nodes. For more information, see - [Launching Alces Flight on AWS using a CloudFormation template](#).

How much will it cost?

The cost for running your cluster will depend on a number of different factors including the resources you consume and the software you subscribe to. 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.)
- Any costs for running the version of Alces Flight that you subscribe to

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.

Flight Compute cluster (Community Support)
Sold by: Alces Flight Ltd

Alces Flight Compute provides a personal, auto-scaling High Performance Computing (HPC) environment for research and scientific computing. Compatible with both on-demand and spot instances, Flight rapidly delivers a whole HPC cluster, ready to go and complete with job scheduler and applications. Clusters are deployed in a Virtual Private Cloud (VPC) environment for security, with SSH and graphical-desktop connectivity for users. Data management tools for POSIX and S3 object storage are also included to help users transfer files and manage storage resources.

Customer Rating Be the first to review this product

Latest Version 2016.2

Operating System Linux/Unix, CentOS 7.2

Delivery Methods
Single AMI
 64-bit Amazon Machine Image (AMI) ([learn more](#))
 Single box deployment of the product
Personal HPC compute cluster
 CloudFormation template ([view](#))
 1 x on-demand login node plus a choice of compute nodes

Support [See details below](#)

AWS Services Required Amazon CloudFormation, Amazon EC2, Amazon EBS

Highlights

- Self-configuring High Performance Compute environment - jump straight to the science instead of configuration.
- Access to the Alces Gridware application repository; providing simple installation of over 750 Linux applications, accelerated libraries and compilers.
- Traditional High Performance Compute cluster look and feel

Pricing Details

For region: **US East (N. Virginia)**

Delivery Methods

☒ Single AMI
☐ Personal HPC compute cluster

Hourly Fees
Total hourly fees will vary by instance type and EC2 region.

EC2 Instance Type	EC2 Usage	Software	Total
t2.large	\$0.104/hr	\$0.00/hr	\$0.104/hr
m4.xlarge	\$0.239/hr	\$0.00/hr	\$0.239/hr
m4.2xlarge	\$0.479/hr	\$0.00/hr	\$0.479/hr
m4.4xlarge	\$0.958/hr	\$0.00/hr	\$0.958/hr
m4.10xlarge	\$2.394/hr	\$0.00/hr	\$2.394/hr
c4.large	\$0.105/hr	\$0.00/hr	\$0.105/hr
c4.2xlarge	\$0.419/hr	\$0.00/hr	\$0.419/hr
c4.4xlarge	\$0.838/hr	\$0.00/hr	\$0.838/hr

Continue You will have an opportunity to review your order before launching or being charged.

Click on the **Continue** button to view details on how to launch.

Launching a Personal HPC cluster from AWS Marketplace

Follow these instructions to launch your cluster:

- After clicking the **Continue** button from the main product page, select the **Custom Launch** tab in your browser.
- Scroll down the page and select your local AWS region in the **Select a Region** section
- Choose **Personal HPC compute cluster** from the *Deployment Options* section
- Under the *Launch* section, click on the **Launch with CloudFormation Console** button to start deploying your cluster.

Launch on EC2:

Flight Compute cluster (Community Support)

1-Click Launch
Review, modify, and launch

Custom Launch
CloudFormation, EC2 Console, APIs or CLI

Launching Options

- You can click "Launch with CloudFormation Console" button below and follow the steps in the CloudFormation console to launch a stack of this software.
- If you prefer to launch just an AMI, you can select Single AMI and click the "Launch with EC2 Console" and follow the instructions to launch an instance of this software.
- If you want to only launch the AMI, you can also find and launch these AMIs by searching for the AMI IDs (shown below) in the "Community AMIs" tab of the EC2 Console [Launch Wizard](#), or launch with the EC2 APIs [Launch Wizard](#).
- You can view this information at a later time by visiting the Your Software page. For help, see [step-by-step instructions](#) for launching Marketplace Products from the AWS Console.

2016.2, released 05/27/2016

[Usage Instructions](#)

Select a Region

EU (Ireland)

Deployment Options

☐ Single AMI
Single Amazon Machine Image (AMI) (learn more)
Single box deployment of the product

☒ **Personal HPC compute cluster**
CloudFormation template (view)
For on-demand login nodes, a choice of compute nodes

Launch

[Launch with CloudFormation Console](#)

Security Group

The vendor recommends using the following security group policies. You will be able to select these settings or configure your own when launching this software.

Connection Method	Protocol	Port Range	Source (IP or Group)
SSH	tcp	22 - 22	0.0.0.0/0
	tcp	5900 - 5919	0.0.0.0/0

Release Notes

2016 Second Release

Pricing Details

For Region
EU (Ireland)

Delivery Methods
Personal HPC compute cluster

Estimated Price

Estimated Software : **Free**

Estimated infrastructure : **\$935/month**

- Estimated infrastructure costs are a typical default deployment with 24x7 usage with the following assumptions
 - EC2 # 1 : 1 x r3.2xlarge machines or equivalent
 - EC2 # 2 : 4 x c4.large machines or equivalent
 - EBS : 4 x 20 GB Compute Disk Magnetic & 1 x 500 GB Shared Disk Magnetic
- Total hourly fees will vary by instance type and EC2 region

As well as an Amazon Machine Image (AMI), Flight Compute subscribers are provided with a CloudFormation template (CFN template) 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 CFN templates to provide more customised environments for specialised requirements. This documentation is designed to assist new users when launching with the CFN template provided on the AWS Marketplace page.

How to answer CloudFormation 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) keeps things more interesting. Avoid using spaces and punctuation, or names longer than 16 characters.

Access and security

- **Cluster administrator username**; enter the username you want to use to connect to the cluster. Flight will automatically create this user on the cluster, and add your public SSH key to the user.
- **Cluster administrator 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.
- **Access network address**; 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.

Alces Flight configuration and customization

- **HPC job scheduler**; select from a range of popular batch job schedulers to install and configure for use with your Alces Flight Compute environment
- **Preload software**; select an [Alces Gridware Depot](#) to install - Alces Gridware Depots are groups of packages, libraries and compilers commonly used by different disciplines
- **Additional features to enable**; optionally select from available [Alces Flight features](#) including job schedulers and other useful customisations
- **S3 bucket for customization profiles**; enter the names of [customisation profiles](#) to use, separated by spaces. Leave this option blank if you have no existing customisation data, or you are starting a standard cluster.
- **Customization profiles to enable**; 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.

Login node

- **Login node instance type**; 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.
- **Specific login node instance type**; if you did not choose a login instance type from the available instance types and chose `other` - you may select from a list of all of the currently available AWS instance types

Compute estate

- **Compute instance type**; 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 available instance types 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
- **Specific compute instance type**; if you did not choose a compute instance type from the available instance types above, and chose `other` - you may select from a list of all of the currently available AWS instance types
- **Spot price**; 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.
- **Autoscaling policy**; select from either `enabled` or `disabled` in this box to enable or disable auto-scaling of your cluster compute nodes
- **Initial compute nodes (autoscaling)**; enter the number of compute nodes you want to start immediately when you have enabled the autoscaling feature. Flight Compute will add more nodes when jobs are queued, and shutdown idle nodes when they have no jobs to process. This parameter is ignored if autoscaling is disabled
- **Initial/maximum compute nodes**; enter the maximum number of compute nodes you wish to make available to your Flight Compute cluster when autoscaling is enabled - the autoscaling feature will never create more than the maximum number specified. If the autoscaling feature is disabled, enter the total number of compute nodes you wish to create at launch time

Disks and storage

- **Data volume layout**; select from a range of data volume layouts - the data volume layouts available are as follows;

standard Configures the home directory share and application directory share using the login node system disk

discrete.home Configures the home directory share on a dedicated EBS volume and application directory share using the login node system disk

discrete.apps Configures the home directory share using the login node system disk and the application directory share using a dedicated EBS volume

discrete.home-discrete.apps Configures both the home directory share and application directory share using separate, dedicated EBS volumes

- **Data volume encryption policy**; if any of the `discrete` options were selected, you may optionally set an encryption policy for the dedicated EBS volumes
- **Login node system disk size**; choose the size of your login node disk, which acts as the shared filesystem for your cluster when using the `standard` data volume layout
- **Login node system volume disk type**; select the [type of EBS volume](#) best suited to your workload requirements - choosing an SSD type will be considerably faster, but choosing a HDD type will incur less running cost

- **Home volume disk size;** if the appropriate data volume layout was chosen to deploy a dedicated home directory EBS volume, you may select the size of volume to deploy
- **Application volume disk size;** if the appropriate data volume layout was chosen to deploy a dedicated application directory EBS volume, you may select the size of the volume to deploy
- **Home volume disk type;** if the appropriate data volume layout was chosen to deploy a dedicated home directory EBS volume, you may choose from a range of EBS volume types for the home directory volume
- **Application volume disk type;** if the appropriate data volume layout was chosen to deploy a dedicated application directory EBS volume, you may choose from a range of EBS volume types for the application directory volume
- **Compute node system disk type;** you may optionally select a system disk type for any deployed compute hosts, allowing you to optimise compute hosts' local ephemeral storage to your workload requirements

The screenshot shows the 'Specify Details' page of an AWS CloudFormation stack. The URL in the browser is `https://eu-west-1.console.aws.amazon.com/cloudformation/home?region=eu-west-1#/stacks/new`. The page title is 'Specify Details'. Below the title, there is a description: '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.](#)'

The 'Stack name' field is filled with 'alces'.

The 'Parameters' section is expanded, showing 'Access and security' parameters:

- Cluster administrator username:** 'alces' (Username for the cluster administrator account).
- Cluster administrator keypair:** A dropdown menu with 'Search' selected (AWS key for access to the cluster administrator account).
- Access network address:** An empty text field (An IP network range (CIDR) that is permitted to access the cluster login node; e.g. entering 0.0.0.0/0 will allow anyone to login using the AWS key specified).

The 'Alces Flight configuration and customization' section is also expanded, showing:

- HPC job scheduler:** A dropdown menu with 'gridscheduler' selected (HPC job scheduler to configure for use within this cluster).

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 CloudFormation 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 CloudFormation 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 CloudFormation template, along with the username you must use to login with your keypair.

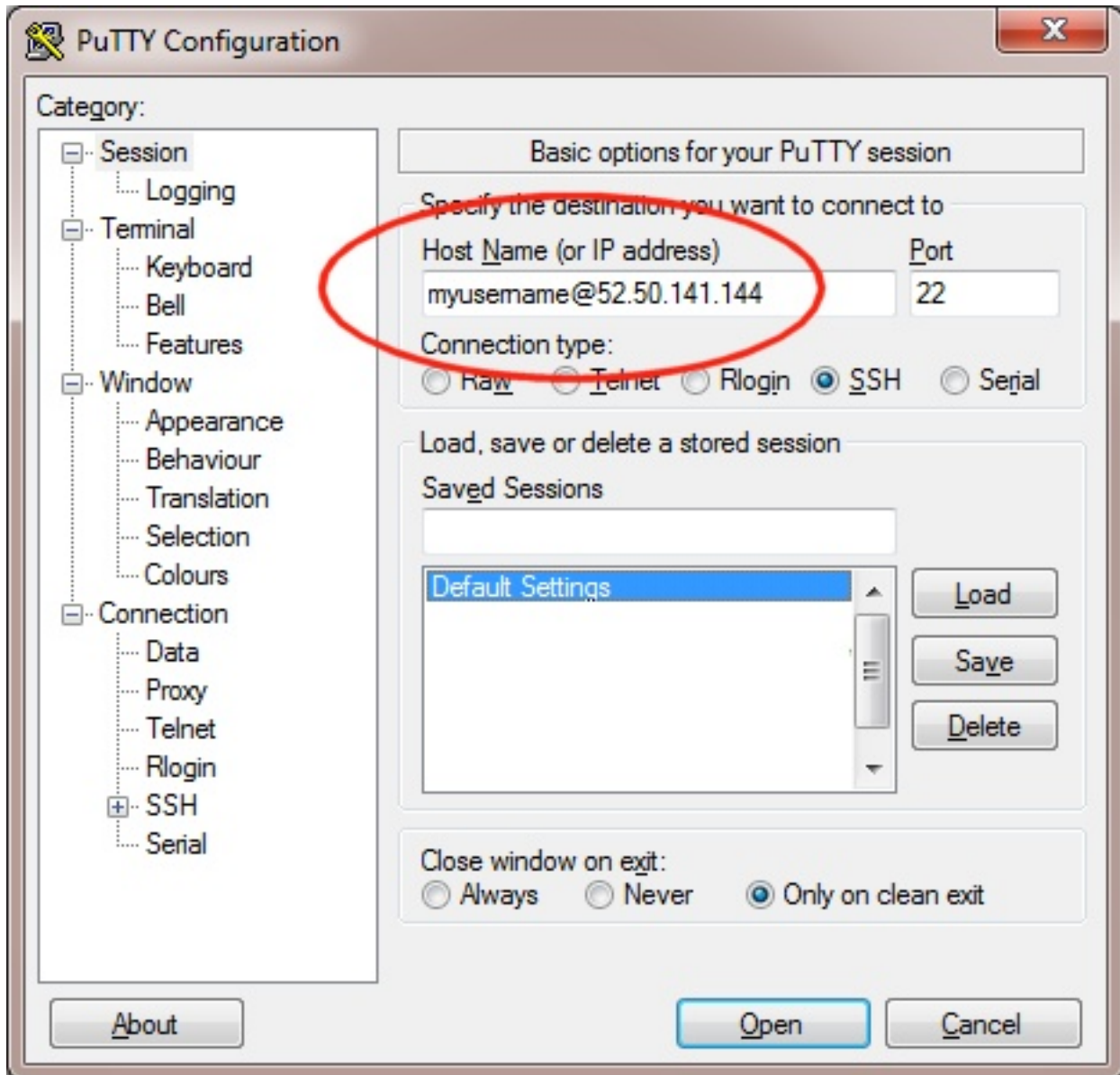
To access the cluster login node from a Linux or Mac client, use the following command:

- `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:



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.


```

alces@login1:~
Using username "alces".
Authenticating with public key "imported-openssh-key"
`.:+//
./oooo+`
`/ooooooo.
/ooooooo/
ooooooo- ./o/
+oooooo/`+ooo
-ooooooo
:ooooooo. `:+:~
-+ooooooo+:ooo`
`:ooooooo.
`:+ooooooo+`
`-+ooooooo+-
.:+ooooooo+-`
`-/ooooooo/..-...-:/+ooo//oooo+/+ooooo/
./ooooooo+ooooooo+ooooooo+ooooooo+`
.+ooooooo+ooooooo+ooooooo+..
.:/+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 [CloudFormation 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* and *alces-sync* for more information on storing your data prior to terminating your cluster.

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. As different clouds can run different versions and configurations of OpenStack, you may need to build images for Flight Compute that suit your platform - alternatively, please contact Alces to purchase consultancy time to build custom images for you.

Prerequisites

- An Alces Flight image for your platform
- An OpenStack user account
- A client machine, correctly connected to the cluster DMZ network
- An appropriate resource quota to create the number of compute nodes you choose
- An OpenStack keypair added for your user

How to deploy

1. Log in to the OpenStack Horizon interface with your site credentials
2. Navigate to the Project -> Orchestration -> Stacks page
3. Select the Launch Stack button
4. When prompted for the **Template Source**, select URL and enter the following template URL:

```
https://raw.githubusercontent.com/alces-software/flight-appliance-support/master/openstack-heat/templates/flight-compute.yaml
```
5. Click the **Next** button to continue

How to answer Heat Stack questions

- The **Stack Name** should be entered with a suitable stack name, this is a unique name within the OpenStack environment and also defines your cluster name
- In the **Creation Timeout** field - a sensible time in minutes should be entered, this stops stack creation after a certain time if the OpenStack environment is too busy to create your resources
- In the **Password for user** field - enter your OpenStack password for the user you are currently logged in as
- In the **Cluster admin key** - select your previously created OpenStack key pair, this is used to log in to the cluster administrator account
- In the **Administrator username** field, enter your chosen administrator username such as `alces`
- In the **Flight Compute image** field, select the Flight Compute image you wish to use - we recommend the latest possible version installed on your system
- Select the compute node type you wish to deploy from the **Compute node instance type** field

- In the **Number of compute nodes** field, enter the number of compute nodes you wish to initially deploy. Note
 - sufficient resource quota must be available in order to launch the nodes

Accessing your cluster

Once your cluster has been launched, the login node will be accessible via SSH. Your login node's access IP address is reported by the OpenStack Heat, along with the username you must use to login with your keypair.

From the Overview tab of your stack, make a note of the cluster master node public IP address displayed, e.g. 10.77.0.100

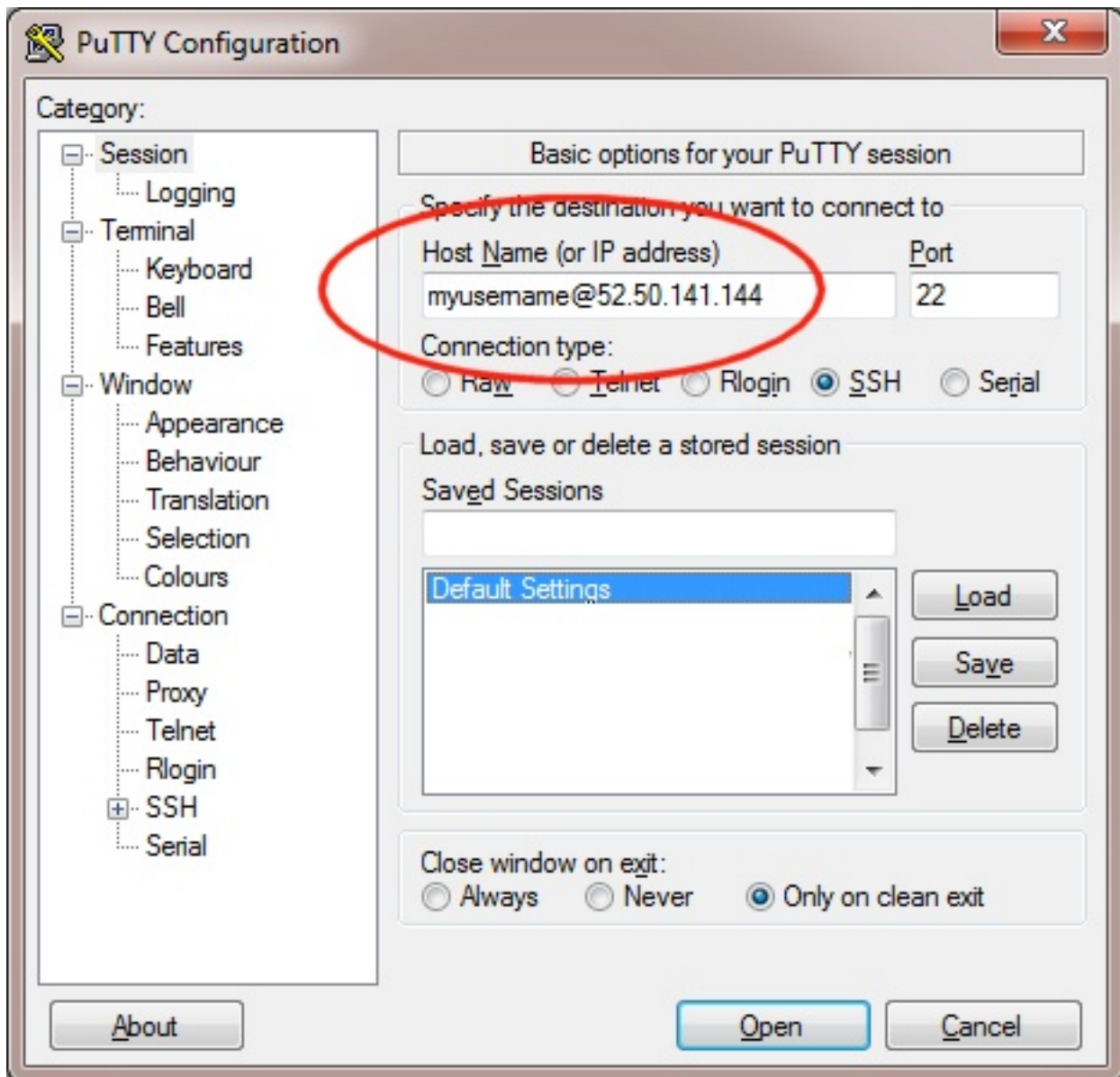
To access the cluster login node from a Linux or Mac client, use the following command:

```
ssh -i mypublickey.pub myadminusername@10.77.0.100
```

Where:

- `mypublickey.pub` is the name of your public SSH key you selected when launching the cluster
- `myadminusername` is the username you entered when launching the cluster
- `10.77.0.100` is the IP address displayed in the Outputs tab of your Heat stack

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:



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.

```
alces@login1:~  
Using username "alces".  
Authenticating with public key "imported-openssh-key"  
`.:./+/  
. /oooo+`  
`./oooooo. Welcome to scooby  
/ooooooo/  
oooooooo- ./o/ Alces Clusterware (r2016.2)  
+oooooo/`+ooo Based on CentOS Linux 7.2.1511 (Core)  
-ooooooooooooo  
:oooooooooooo. `:+: `  
-+oooooooooooo+:ooo`  
`:oooooooooooo.  
`:+oooooooooooo+` /o/  
`-+oooooooooooo+- .. .+/ -ooo:  
..:+oooooooo+- .+o: `:ooo :oooo+  
`-/ooooooooo/-.-....-:/+ooo//oooo+/+ooooo/  
./oooooooooooo+ooooooooooooooooooooooooooooo/+`  
.+oooooooooooo++oooooooooooooooooooo+.:.  
../+oooooooo-`.--:::-:-.`  
`-:/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 your environment

When you terminate your environment, your cluster login and compute nodes will be terminated. Any data held on attached storage may be erased, with storage volumes being wiped and returned to the shared 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.**

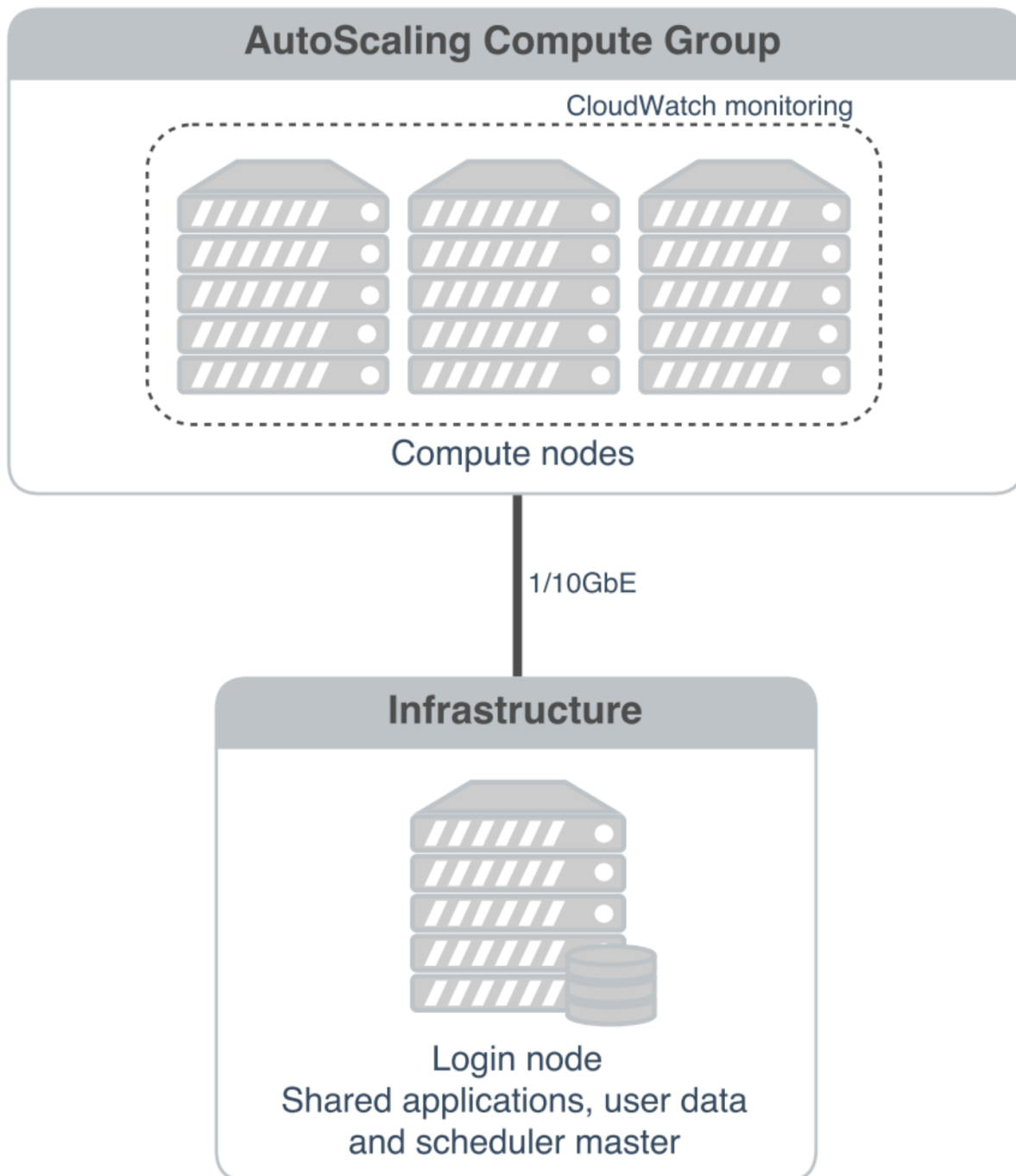
1. From the `Stacks` page, select your previously created Alces Flight Compute stack - then select `Delete Stacks`

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 CloudFormation 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.

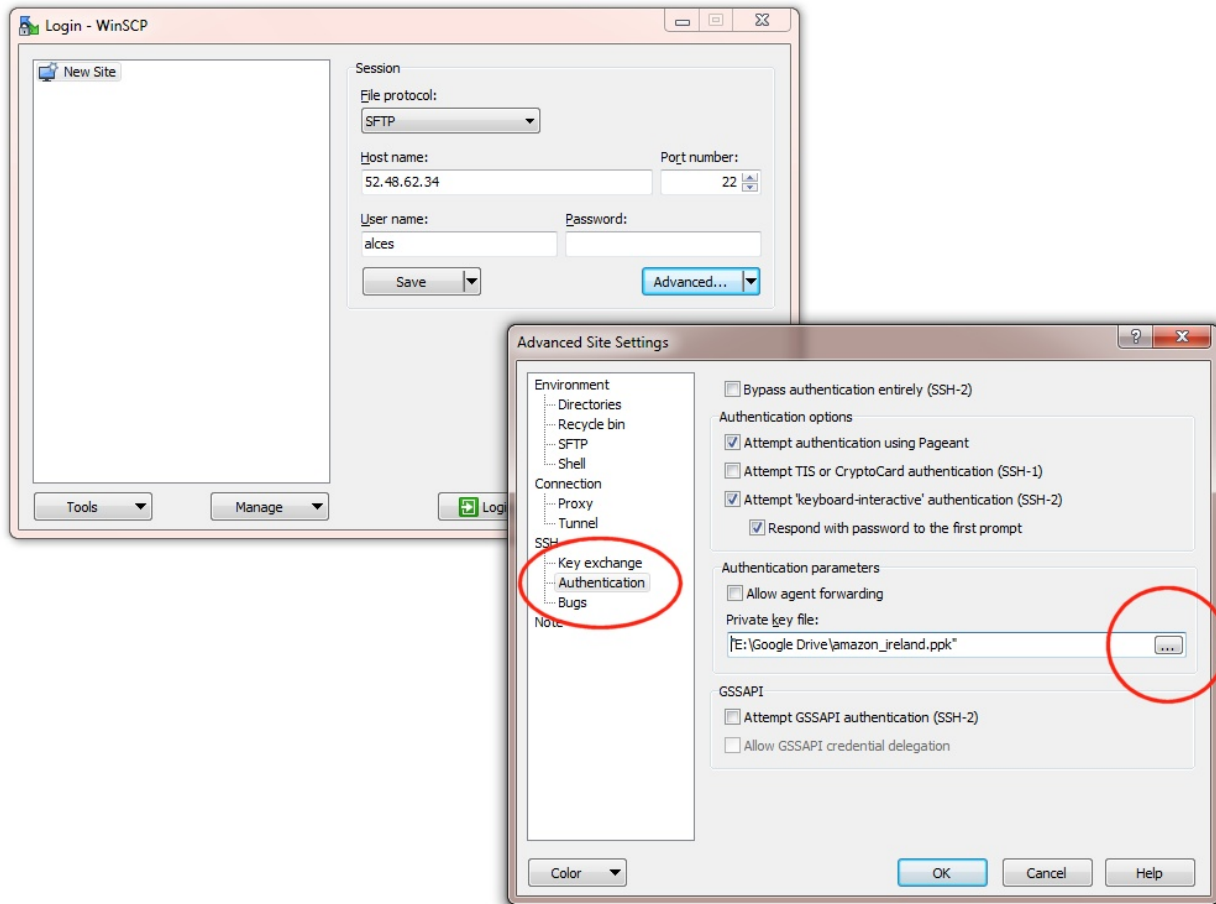
- **To copy file `myresults.zip` from your cluster on IP address `52.48.62.34` to your local Linux or Mac client:** `scp -i mykeyfile.pub jane@52.48.62.34:/home/jane/myresults.zip .`

Using a graphical client to copy data

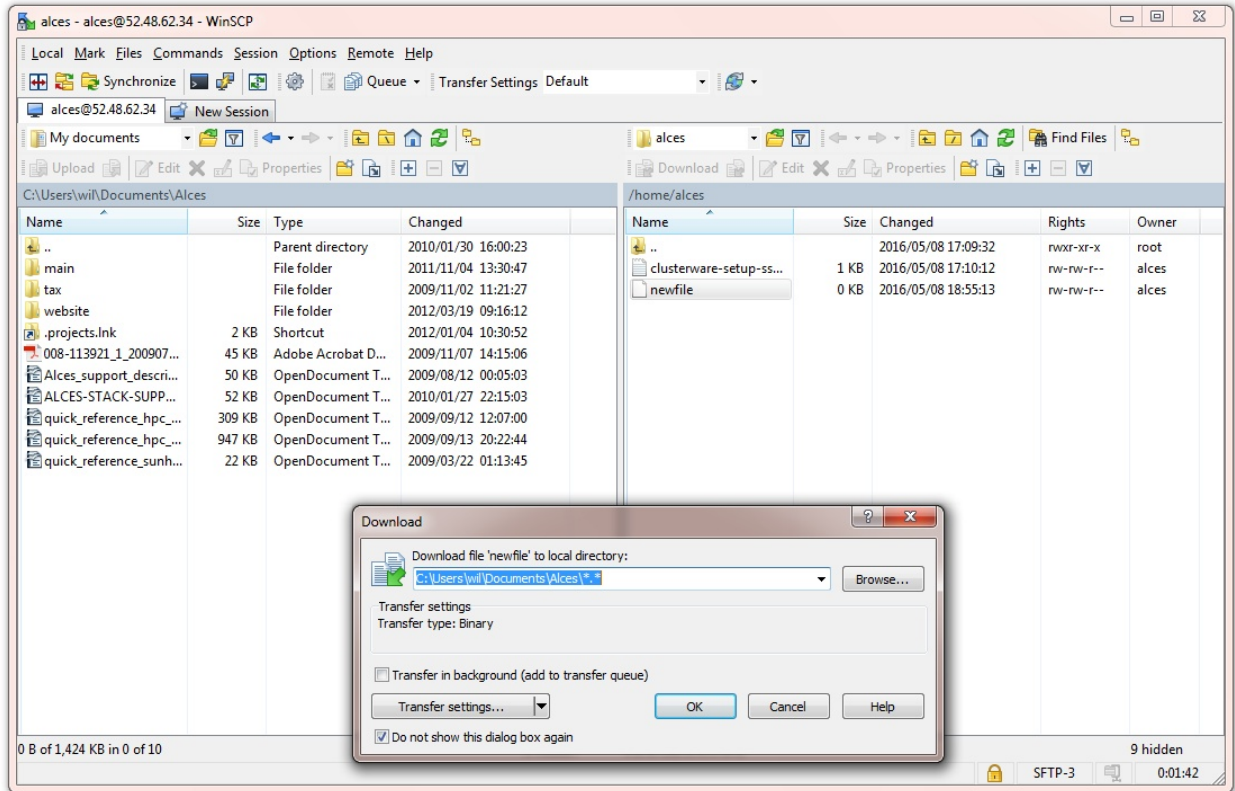
There are also a number of graphical file-management interfaces available that support the SSH/SCP/SFTP protocols. A graphical interface can make it easier for new users to manage their data, as they provide a simple drag-and-drop interface that helps to visualise where data is being stored. The example 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.



- 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



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 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 independently 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**, **Swift** and **Dropbox** accounts. Object storage services which are compatible with S3 or Swift can also be configured. For example - a Ceph storage platform with a compatible **RADOS-gateway** can 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 swift` - enables **Swift** 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-s3areal s3
Display name [my-s3areal]:
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.

To configure access to a Swift compatible storage service, enter your username, API key and endpoint URL for the service. Please contact your storage service administrator to obtain these values; e.g.

```
[alces@login1(scooby) ~]$ alces storage configure my-swift swift
Display name [my-swift]:
Username: SLOS9275161
API key: *****
Authentication endpoint: https://lon02.objectstorage.softlayer.net/auth/v1.0/
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-s3areal
alces storage use: storage configuration 'my-s3areal' 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) ~]$
```

Note: As well as being able to recursively put entire directories from a local path into the remote storage target, users can also get and rm directories recursively, again using the `-r` or `-R` option with their `alces storage` command.

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.

Home Directory Synchronization

Alces Flight Compute is designed to provide users with a personal, ephemeral environment - allowing you to work when you need to, and shut down your cluster to free up resources when you're done. Flight includes a range of tools and utilities to help you setup your cluster quickly and easily, minimising the time taken to launch a new cluster and make it ready for use. The `alces-sync` utility helps users to manage the files and data you may typically store in your home directory such as job scripts and output data. The tool allows users to synchronise new clusters using data stored online in an S3 bucket, and to quickly and easily store their data when they've finished using a cluster before shutting it down.

Getting started

When logging in to your Alces Flight Compute for the first time - the Alces synchronization tool will pull any data stored in the default location in your S3 bucket. For an AWS account that has either not used the Alces synchronization utility or not created an Alces Flight Compute environment before - no files will need to be synchronized.

To view the synchronization location of your current user, you can use the following command to displays both the local directory to be synced and its remote S3 location:

```
[alces@login1(scooby) ~]$ alces sync list
default: /home/alces <-> s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default
```

Note: The `alces sync` utility *does not* automatically push files and data back to the remote synchronization target, please view the documentation on how to push data back to the remote synchronization target.

Pushing data back to the remote location

The `alces sync` utility includes a tool to push any new data back to the remote synchronization target. The following example shows the process of adding some files and directories in your home directory, then pushing them back to the remote location. The command to use is `alces sync push` - which synchronizes any data in your home directory back to the remote synchronization target.

```
[alces@login1(scooby) ~]$ mkdir job-scripts
[alces@login1(scooby) ~]$ touch job-scripts/imb_2node.sh
[alces@login1(scooby) ~]$ mkdir outputs
[alces@login1(scooby) ~]$ for i in `seq 1 5`; do touch outputs/$i; done
[alces@login1(scooby) ~]$ alces sync push

> Synchronizing directory '/home/alces' to s3://alces-flight-nmi0ztdmyzm3ztm3
Permissions ... OK
Encryption passphrase (CTRL+C to skip): Skipping.
Sync ... OK
```

You can then verify that the synchronization was successful either using the S3 web console, or via the alces storage utility when configured for your S3 target:

```
[alces@login1(scooby) ~]$ alces storage ls s3://alces-flight-nmi0ztdmyzm3ztm3/sync/
↪alces/default/

DIR    s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.config/
DIR    s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.ssh/
DIR    s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/job-scripts/
DIR    s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/outputs/
2016-09-06 13:52      358  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪bash_history
2016-09-06 13:47        18  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪bash_logout
2016-09-06 13:47      193  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪bash_profile
2016-09-06 13:47      231  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪bashrc
2016-09-06 13:47      334  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪emacs
2016-09-06 13:47      887  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪modulerc
2016-09-06 13:47      740  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/.
↪modules
2016-09-06 13:47      118  s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default/
↪clusterware-setup-sshkey.log
```

Hint: When performing the `alces sync push` command - you may choose to set an encryption passphrase for additional protection.

Managing pushed files and data

You may not wish to push *all* of the files and data from a local synchronization target, particularly when working with your home directory which contains many one-time-use configuration files which may cause issues with any later deployed environments.

Your home directory contains a Flight Compute configuration management folder, located at:

```
~/.config/clusterware/
```

Inside the configuration directory - there will be a configuration file for each synchronization target - for example:

```
sync.data.yml Configuration file for the data synchronization target
```

```
sync.default.yml Configuration file for home directory synchronization
```

Open the `sync.default.yml` configuration file to add an example file exclusion. Below the `:source: :home` line - include a new section - e.g.

```
---
:source: :home
:exclude:
- ".modules"
- ".ssh/id_*
```

The above example would prevent the `.modules` file as well as any file matching the wildcard search `id_*` in the `.ssh` directory from being pushed to the remote synchronization target.

Encrypting data

The `alces sync` tool offers the choice to optionally set an encryption key on any uploaded data. This provides an extra layer of security for the data stored in your S3 buckets. A secure and most importantly memorable passphrase should be used - failing to remember your passphrase will prevent you from obtaining or using the data later on.

When performing the `alces sync push` command - you will be prompted for an encryption passphrase, or optionally given the choice to skip using a passphrase by pressing **Ctrl+C**.

A passphrase can contain any form of characters, just ensure the passphrase is either remembered or stored in a secure location for retrieval and access later on.

The following example shows the `alces sync push` command being used, setting an encryption passphrase on the uploaded data:

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

> Synchronizing directory '/home/alces' to s3://alces-flight-nmi0ztdmyzm3ztm3
  Permissions ... OK
Encryption passphrase (CTRL+C to skip):
  Sync ... OK
```

When the `alces sync` tool retrieves data, either automatically on first login or manually via the `alces sync pull` command - you will be prompted to enter your encryption passphrase if one was previously set. Without the encryption passphrase, you will not be able to obtain or use your stored data.

Pulling files and data

If you have pushed any new data from another running Alces Flight Compute environment, or manually uploaded files to the remote synchronization target - you may wish to pull those files to your current Alces Flight Compute environment. The following command displays the `alces sync pull` command retrieving remote data from the synchronization target:

```
[alces@login1(scooby) ~]$ ls
clusterware-setup-sshkey.log

[alces@login1(scooby) ~]$ alces sync pull
> Synchronizing directory '/home/alces' from s3://alces-flight-nmi0ztdmyzm3ztm3
  Sync ... OK
  Permissions ... OK

[alces@login1(scooby) ~]$ ls
clusterware-setup-sshkey.log  job-scripts  outputs
```

Adding and removing synchronization targets

The following section details the process of adding and removing additional storage synchronization configurations.

Adding a synchronization configuration

In addition to the default home-directory synchronization configuration, you may wish to set up additional configurations to assist you with commonly used data directories.

The following example demonstrates how to set up a synchronization target for the local `/data` directory:

```
[alces@login1(scooby) ~]$ alces sync add data /data
alces sync add: created 'data' to sync '/data'

[alces@login1(scooby) ~]$ alces sync list
data: /data <-> s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/data
default: /home/alces <-> s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default
```

You can then synchronize any data stored in the `/data` directory using the `alces sync push <name>` command.

Note: By default, the `alces sync push` command will only push data to the default storage configuration - typically the users home directory. It is important to ensure you specify the storage configuration name when using the `alces sync push` command to manage additionally set up storage configurations.

Removing a synchronization configuration

You may wish to remove a storage synchronization configuration from your Alces Flight Compute environment - this can be achieved using the `alces sync remove` command - as demonstrated below:

```
[alces@login1(scooby) ~]$ alces sync list
data: /data <-> s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/data
default: /home/alces <-> s3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/default

[alces@login1(scooby) ~]$ alces sync remove data
Remove sync configuration for 'data' (Y/N)? y
alces sync remove: removed 'data'
```

Wiping remote storage configuration targets

In order to assist you with data management, the `alces sync` utility provides an easy method of removing all files and data stored within a remote synchronization target. To clear the contents of a remote storage target, use the `alces sync purge <name>` command as demonstrated below:

```
[alces@login1(scooby) ~]$ alces sync purge data
Purge all files for 'data' at 's3://alces-flight-nmi0ztdmyzm3ztm3/sync/alces/data' (Y/
↵N)? y
alces sync purge: purged 'data'
```

Warning: Double check the remote location you are planning to wipe **does not** contain any important data before running the `alces sync purge` command.

Graphical desktop access to your login node

Your Alces Flight Compute login node can 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
```

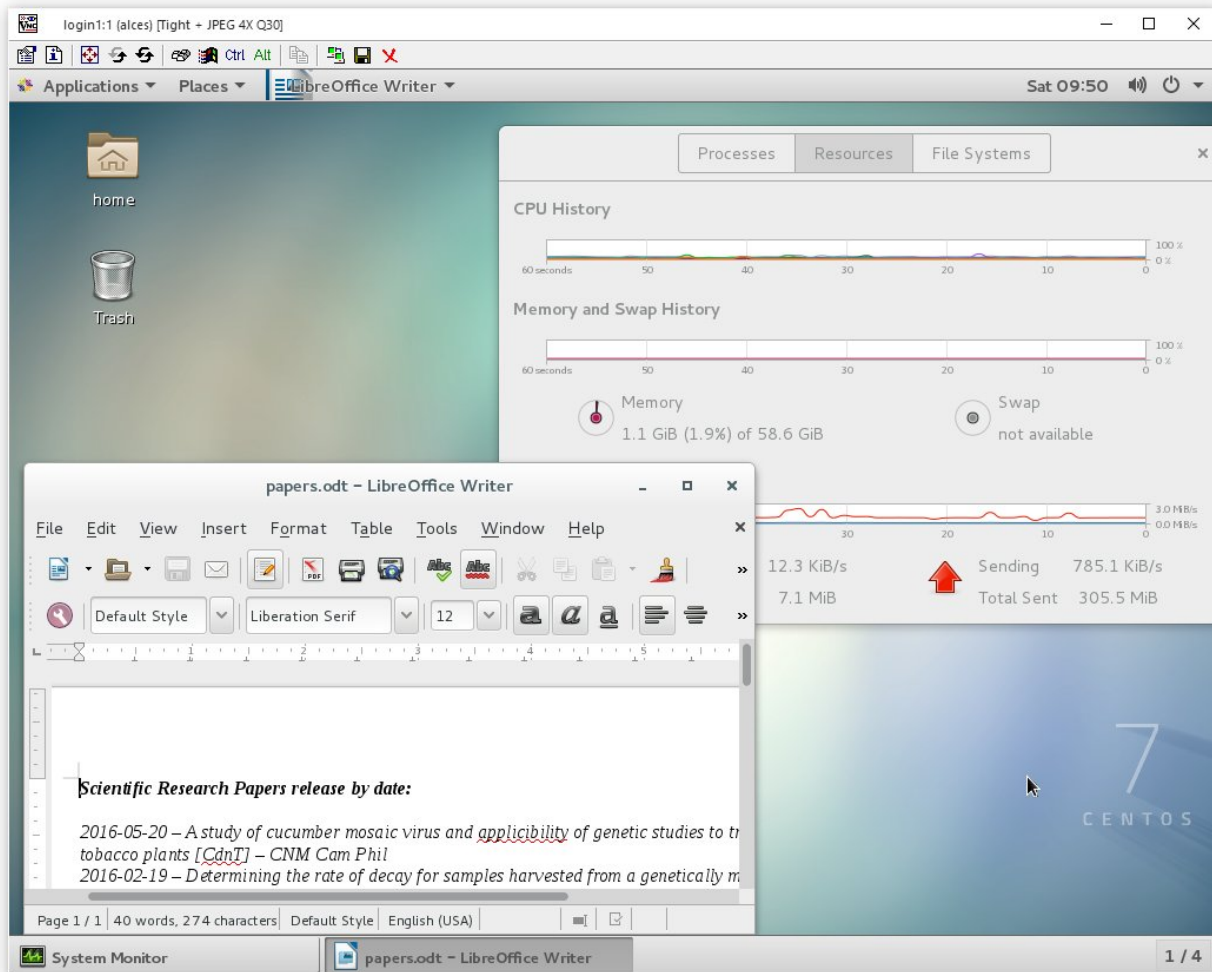
Users need a VNC client to connect to the graphical desktop session - for a list of tested clients, see [What is Alces Flight Compute?](#).

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:



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:



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

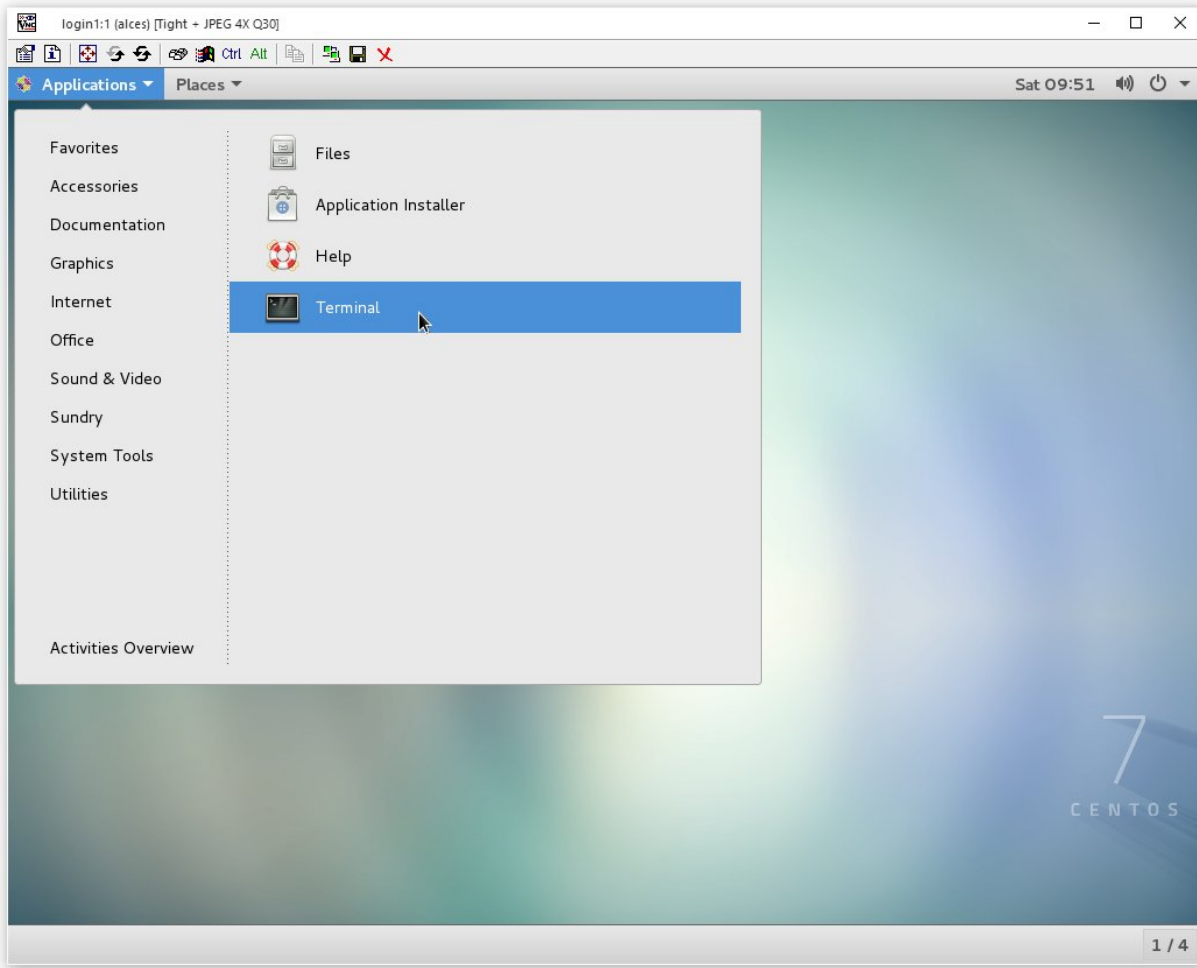
Specifying a size with the `alces session` tool

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

```
alces session start --geometry 1920x1080 gnome
```

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

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



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

```
[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.

Cluster VPN

Your Flight Compute cluster is configured with a Virtual Private Network (VPN) - allowing you to connect your workstation to the cluster network. The cluster VPN provides your machine with an IP address that is part of the environment's network, allowing direct communication with compute nodes for high-performance graphical application access and data transfer. All communications exchanged over a VPN are automatically encrypted using a certificate unique to your cluster.

The Alces Flight Compute VPN uses [OpenVPN](#) - so you may wish to use a client that is capable of connecting using OpenVPN configurations.

Note: For best security practice, each Alces Flight Compute cluster you deploy is configured with a unique VPN configuration. If you run multiple clusters, users need to create a profile on your client machine for each Alces Flight Compute environment you want to connect to.

Available VPN clients

There are many VPN clients available which support OpenVPN configurations - some popular clients for each platform include;

MacOS

- [TunnelBlick](#)

Linux

- NetworkManager includes support for OpenVPN configurations

Windows

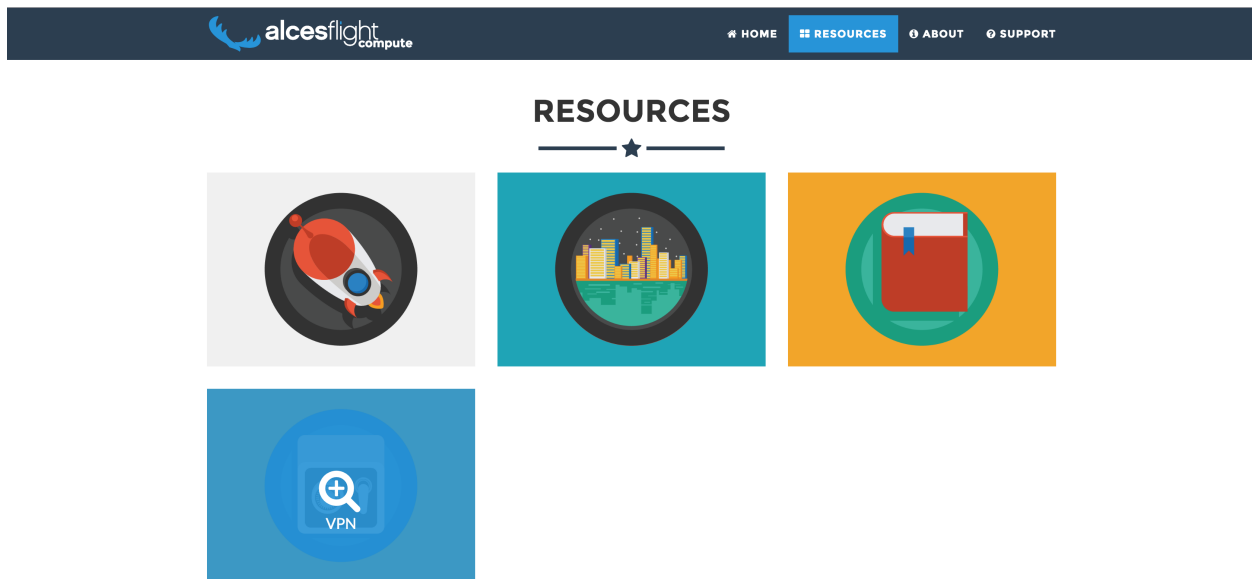
- [OpenVPN](#)

Obtaining VPN configuration

VPN configuration packs for different VPN clients are available through the Alces Flight web page. You can find your Alces Flight web page access information by:

- Viewing the `Outputs` tab of your AWS CloudFormation Flight Compute stack - which displays the `http` access address
- Logging on to your Flight Compute environment and running the command `alces about www` which displays access information

Once you have navigated to your Alces Flight Compute web page - you can click on the **VPN** button, which will allow you to visit the VPN configuration page. From the VPN configuration page, you will be offered a range of VPN configuration file packs for different VPN client types as well as brief instructions on how to connect to the VPN using your downloaded configuration packs:



DOWNLOADS



Choose a configuration archive suitable for your platform from those offered below, or [browse all available downloads](#). Refer to the [information below](#) for how to configure your VPN client, or refer to the [Alces Flight Appliance documentation](#).

Please note: You will need to provide the VPN configuration download access password in order to download these configurations. Find the download access password by running `alces about vpn` at your cluster command line.

TAR ARCHIVE (OPENVPN)

Provides configuration suitable for an OpenVPN client for those familiar with the Tar format. This archive is suitable for users of Linux.

[Download Tar](#)

ZIP ARCHIVE (OPENVPN)

Provides configuration suitable for an OpenVPN client for those more familiar with the ZIP format. This archive is suitable for users of Windows.

[Download ZIP](#)

ZIP ARCHIVE (TUNNELBLICK)

Provides configuration suitable for use with the popular open-source OpenVPN client for macOS, [Tunnelblick](#).

[Download ZIP](#)

Note: You can also use command-line tools such as `scp` to download your VPN configuration packs. Use the command `alces about vpn` on your Flight Compute environment to view the locations of the VPN configuration packs before transferring to your local workstation

Software Applications

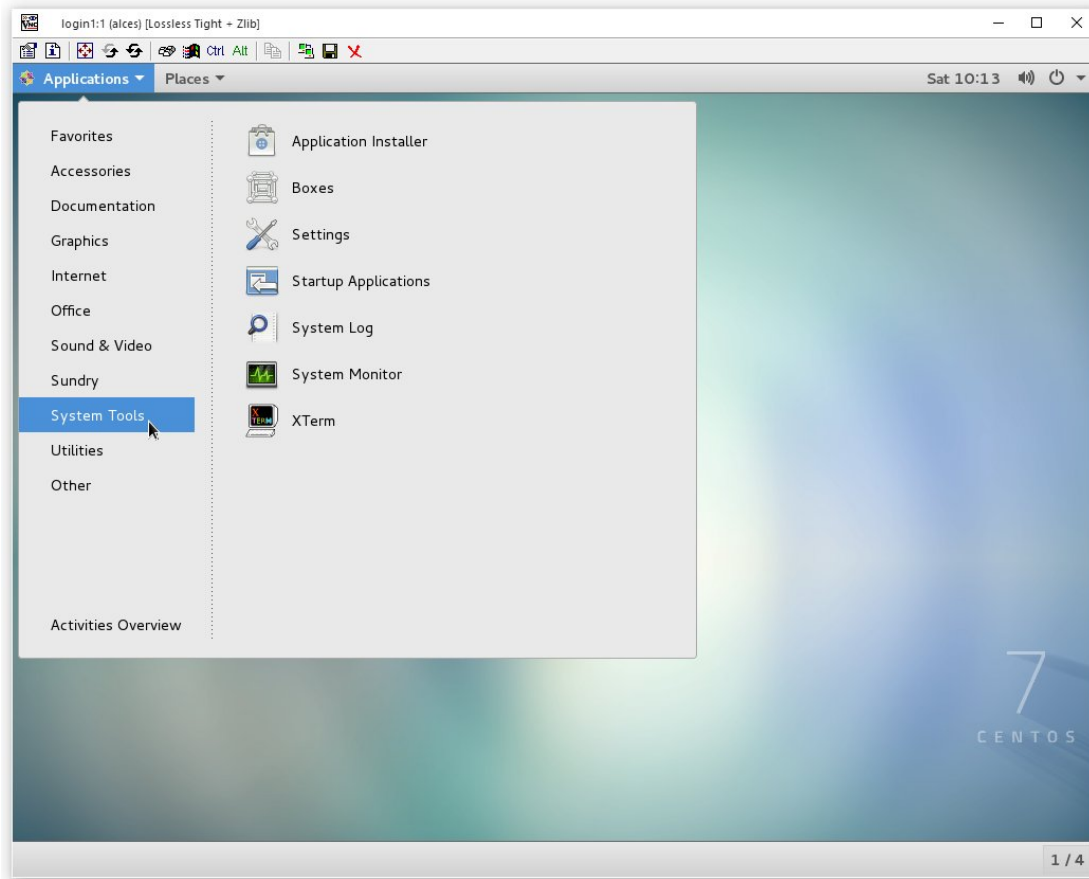
Flight Compute operating system

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.

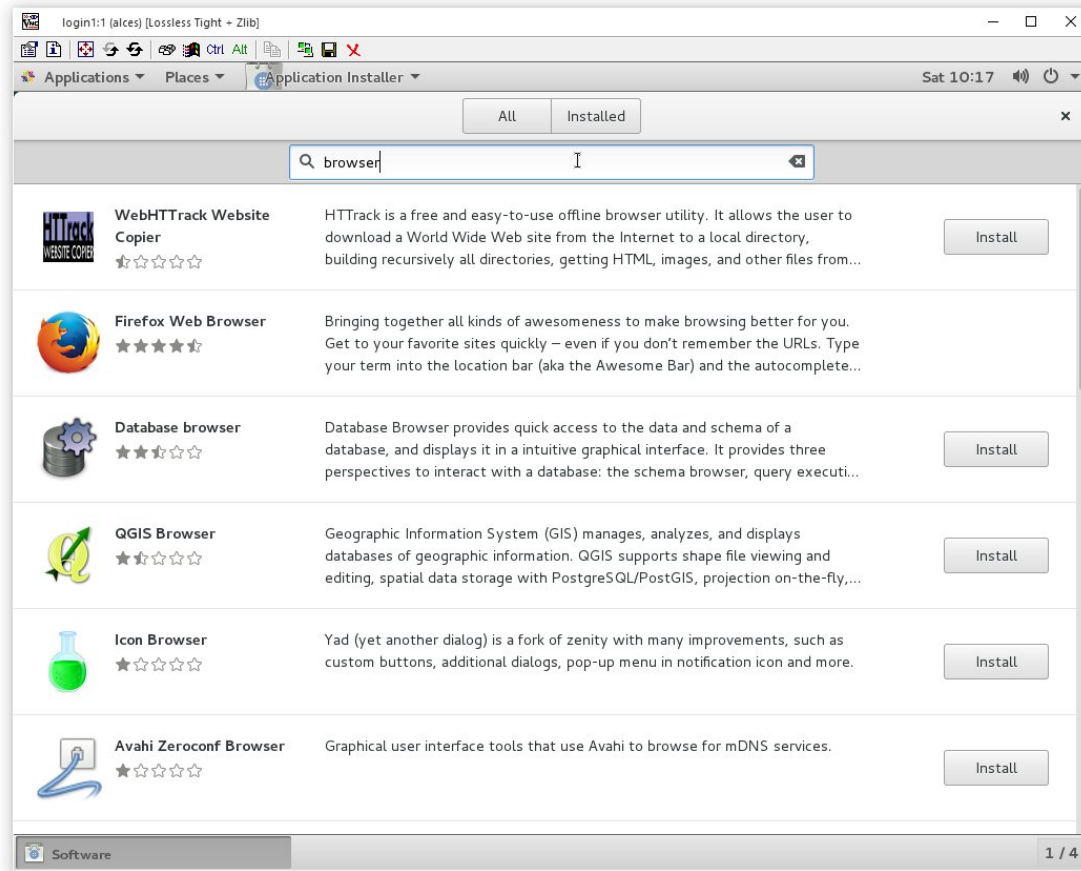
Installing Linux applications on the login node

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 with** `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



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 two 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

Note: Depending on the version of Flight Compute you are using, you may have the option to choose capacity and performance characteristics of the shared applications volume at cluster launch time. Ensure that you choose a large enough storage area to suit the applications you want to install.

Installing cluster applications

Alces Flight Compute clusters include access to the online **Gridware** repository of software applications. This catalogue includes over 850 application, library, compiler and MPI versions which can be installed and run on your cluster. Software is installed by selecting it from the catalogue, optionally 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 your cluster scheduler.

Application catalogue structure

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

- **repository** - packages are listed in the **main** repository if available for auto-scaling clusters, 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

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) ~]$

```

Note: Gridware will automatically install pre-compiled binary versions of applications from the **main** repository, if they are available. Users can optionally use the `--no-binary` parameter to force packages to be compiled at installation time.

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.

Applications that have Linux distribution dependencies will trigger installation of any required packages when their module is loaded on compute nodes for the first time. This allows newly launched nodes (e.g. in an auto-scaling cluster) to automatically resolve and install any dependencies without user intervention.

Note: Automatic dependency installation can occasionally cause a brief delay at module load time when an application is run on a new compute node for the first time.

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 loading 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 support automatic dependancy resolution, enabling support for auto-scaling clusters where compute nodes may be sourced from the AWS spot market. This allows Linux distribution dependancies to be satisfied dynamically at `module load` time, ensuring that software applications execute correctly whenever they are run. 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.

Note: Users installing applications from the `volatile` repo should either ensure that auto-scaling is disabled for their user environment, or make use of Flight customization features to ensure that software package dependancies are resolved for new compute nodes joining the cluster after applications have been installed.

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                Size          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 a depot

Alces Flight Compute clusters also support collated 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)
- 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 superseded 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**, **HPMPI**; commercial MPIs 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.

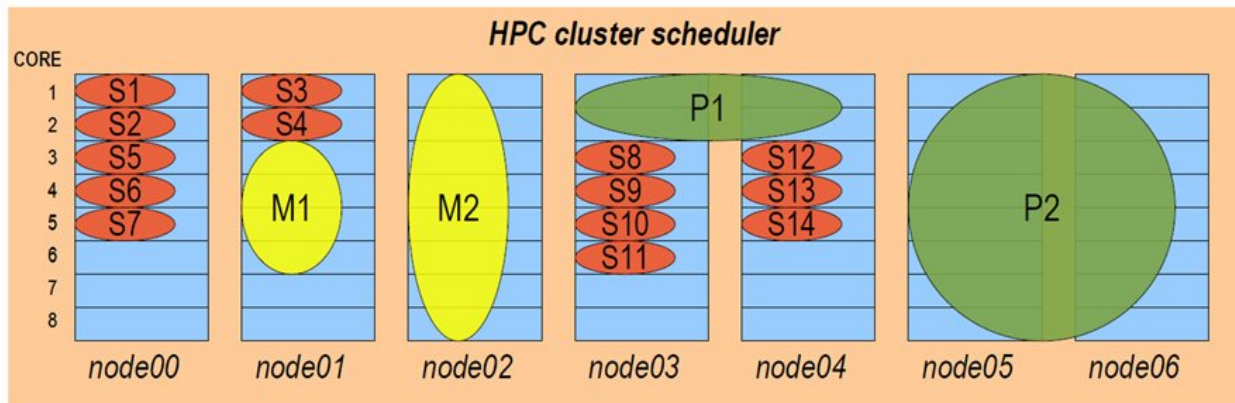
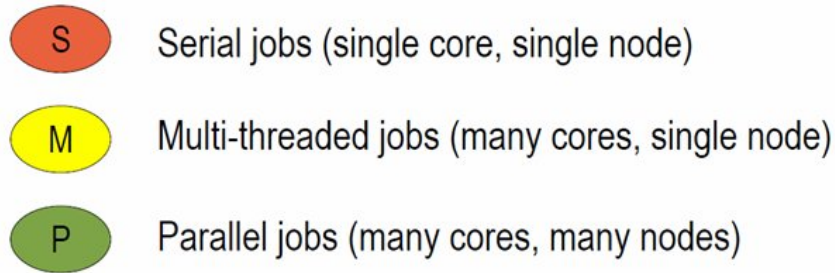
When a new job is submitted by a user, the cluster scheduler software assigns compute cores and memory to satisfy the job requirements. If suitable resources are not available to run the job, the scheduler adds the job to a queue until enough resources are available for the job to run. You can configure 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:



Interactive and batch jobs

Users typically interact with compute clusters by running either **interactive** or **batch** (also known as **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.

Available cluster job schedulers

Alces Flight Compute Community Edition launches with a pre-configured [Open Grid Scheduler \(SGE\)](#) environment - 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.

Alces Flight Compute can optionally be configured to launch the following cluster job schedulers;

- [Slurm scheduler](#)
- [OpenLava scheduler](#) (similar to IBM LSF)
- [Torque scheduler](#)
- [PBS Pro scheduler](#)

Job-scheduler configuration and support

Alces Flight Compute clusters are designed to pre-install your chosen cluster job scheduler, and make it available to use by the single-user configured at launch time. Customers are provided with dedicated support contacts to help out where needed. For full commercial code-base support for job-schedulers, please contact the vendor of your chosen software product.

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 operating system and derivatives should be possible to run on your cluster.

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 `qssh` 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.

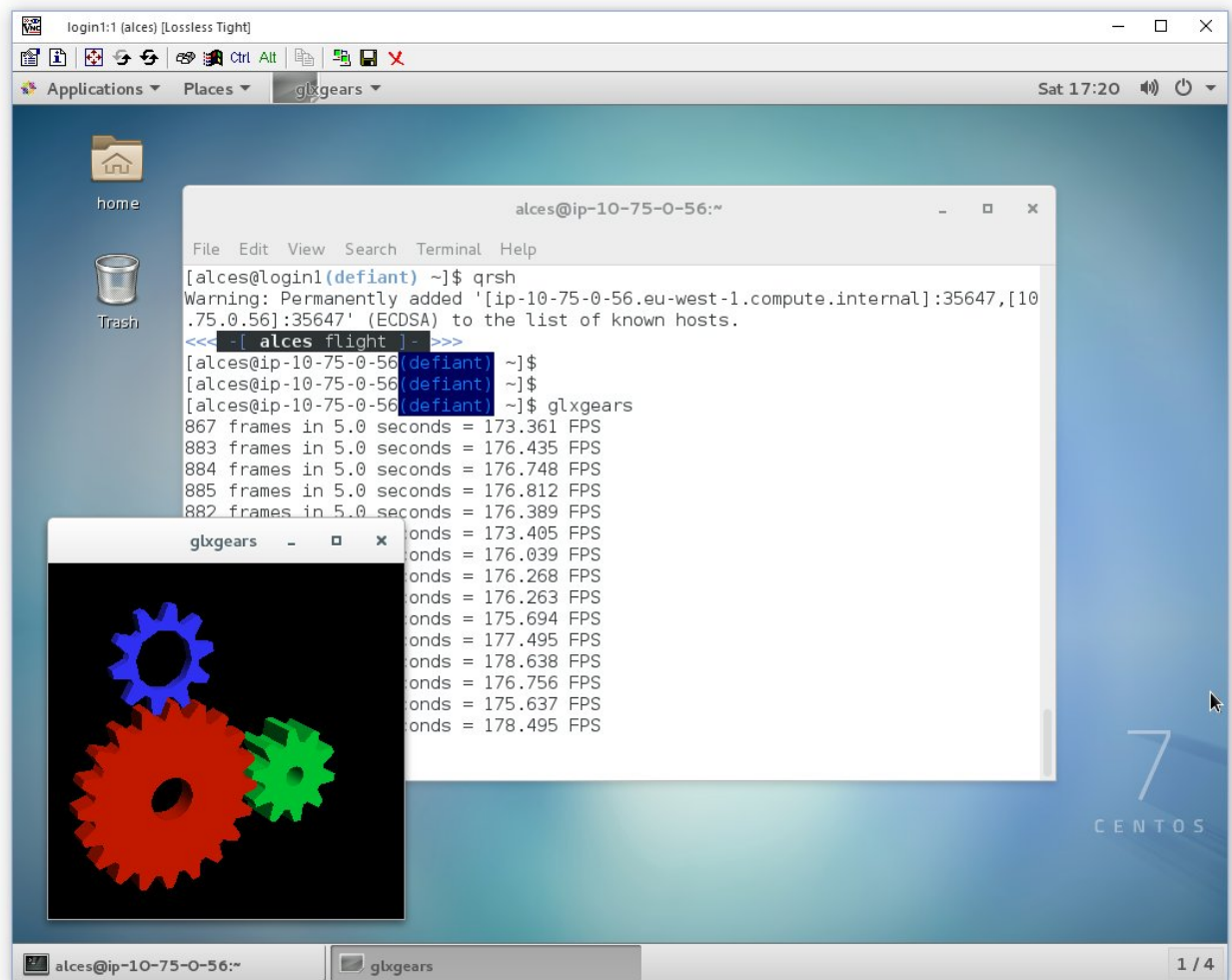
<<< -[ alces flight ]- >>>
[alces@ip-10-75-0-21(scooby) ~]$ hostname -f
ip-10-75-0-21.eu-west-1.compute.internal

[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:



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) ~]$ qysh
Your "qysh" 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 `qysh` 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
-----
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
-----
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 `qghost` 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 - see [Disabling resource limits for your job-scheduler](#) for instructions.

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
↳ 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 **s**ymmetric **m**ulti-**p**rocessing). 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 $4 \times 1.5\text{GB} = \mathbf{6GB}$ 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 $64 \times 1.5\text{GB} = 96\text{GB}$ 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 successfully 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.

Controlling scheduler configuration

Your Alces Flight Compute environment provides tools to quickly configure your cluster scheduler, controlling the job spanning type as well as the job submission strategy.

Viewing the current configuration

You can view the current scheduler configuration using the `alces configure scheduler status` command:

```
[alces@login1(sge) ~]$ alces configure scheduler status
=====
                                gridscheduler
=====
Allocation strategy: packing
Submission strategy: master
```

Scheduler allocation strategy

Changing the scheduler allocation strategy allows you to choose from a variety of different methods of scheduling jobs across available nodes. The currently available configuration types are as follows:

packing Pack as many jobs as possible onto as few compute nodes as possible. This method is useful to minimise the number of separate compute nodes you need in your cluster, and can help to minimise running costs in an environment where users are charged per compute node.

spanning Select the least busy nodes for new jobs. This method is useful to help ensure the best possible performance for each job.

To control the scheduler allocation strategy, use the command `alces configure scheduler allocation <strategy>` as per the following example:

```
[alces@login1(sge) ~]$ alces configure scheduler allocation spanning
alces configure scheduler: gridscheduler: allocation strategy set to: spanning
```

Scheduler submission strategy

The default scheduler submission strategy allows only jobs submitted from the cluster login node to be accepted. You may wish to change this to allow cluster compute hosts to submit jobs, which can be useful for multi-stage workflows - or disable job submission entirely. The current scheduler submission strategies available are:

all Accept job submission from all cluster hosts

master Accept job submission from the login/master node only

none Disable job submission from all cluster hosts

To control the scheduler submission strategy, use the command `alces configure scheduler submission <strategy>` as per the following example:

```
[alces@login1(sge) ~]$ alces configure scheduler submission all
alces configure scheduler: gridscheduler: submission strategy set to: all
```

Slurm Scheduler

The [Slurm](#) cluster job-scheduler is an open-source project used by many high performance computing systems around the world - including many of the [TOP 500](#) supercomputers.

See [Cluster job schedulers](#) for a description of the different use-cases of a cluster job-scheduler.

Running an interactive job

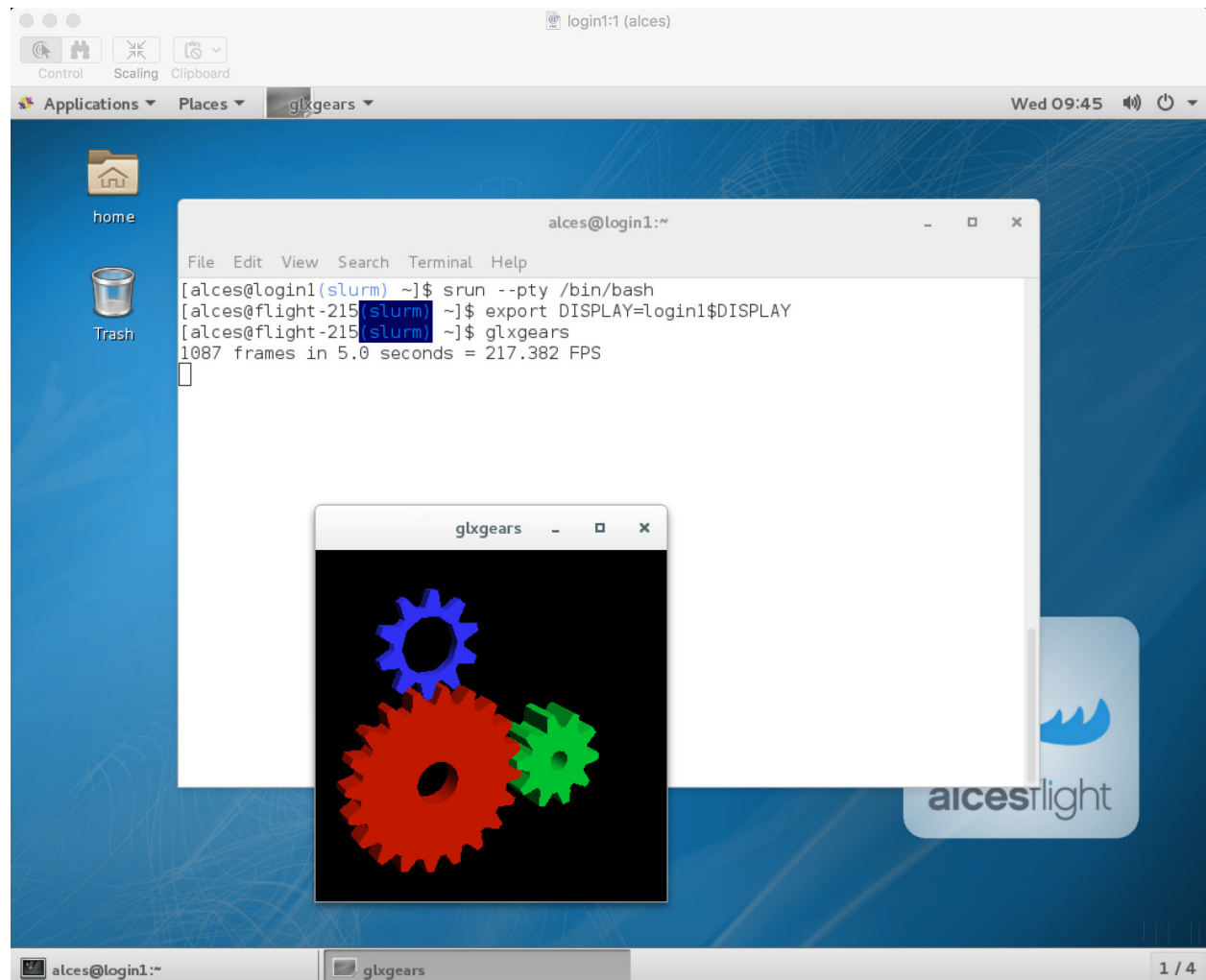
You can start a new interactive job on your Flight Compute cluster by using the `srun` command; the scheduler will search for an available compute node, and provide you with an interactive login shell on the node if one is available.

```
[alces@login1(scooby) ~]$ srun --pty /bin/bash
[alces@ip-10-75-1-50(scooby) ~]$
[alces@ip-10-75-1-50(scooby) ~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
3	all	bash	alces	R	0:39	1	ip-10-75-1-50

In the above example, the `srun` command is used together with two options: `--pty` and `/bin/bash`. The `--pty` option executes the task in pseudo terminal mode, allowing the session to act like a standard terminal session. The `/bin/bash` option is the command that you wish to run - here the default Linux shell, BASH.

Alternatively, the `srun` 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 `srun` session are executed on the assigned cluster compute node.



Note: The Slurm scheduler does not automatically set up your session to allow you to run graphical applications inside an interactive session. Once your interactive session has started, you must run the following command before

running a graphical application: `export DISPLAY=login1$DISPLAY`

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

If the job-scheduler could not satisfy the resource 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) ~]$ srun --pty /bin/bash
srun: job 20 queued and waiting for resources
```

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:

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

To submit your job script to the cluster job scheduler, use the command `sbatch 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) ~]$ sbatch simplejobscript.sh
Submitted batch job 21

[alces@login1(scooby) ~]$ ls
clusterware-setup-sshkey.log  simplejobscript.sh  slurm-21.out

[alces@login1(scooby) ~]$ cat slurm-21.out
Starting running on host ip-10-75-1-50
Finished running - goodbye from ip-10-75-1-50
```

Viewing and controlling queued jobs

Once your job has been submitted, use the `squeue` command to view the status of the job queue. If you have available compute nodes, your job should be shown in the `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 the `PD` (pending) state until compute nodes are available to run it. If a job is in `PD` state - the reason for being unable to run will be displayed in the `NODELIST (REASON)` column of the `squeue` output.

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

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
41	all	simplejo	alces	R	0:03	1	ip-10-75-1-50
42	all	simplejo	alces	R	0:00	1	ip-10-75-1-50

You can keep running the `squeue` 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. You can use the Linux `more` command to view your output file:

```
[alces@login1(scooby) ~]$ more slurm-42.out
Starting running on host ip-10-75-1-50
Finished running - goodbye from ip-10-75-1-50
```

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 `squeue` command to see where they run. The scheduler is likely to spread them around over different nodes (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 `scancel <job-ID>` command to delete a job you've submitted, whether it's running or still in the queued state.

```
[alces@login1(scooby) ~]$ sbatch simplejobscript.sh
Submitted batch job 46
[alces@login1(scooby) ~]$ sbatch simplejobscript.sh
Submitted batch job 47
[alces@login1(scooby) ~]$ sbatch simplejobscript.sh
Submitted batch job 48
[alces@login1(scooby) ~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
43	all	simplejo	alces	R	0:04	1	ip-10-75-1-50
44	all	simplejo	alces	R	0:04	1	ip-10-75-1-50
45	all	simplejo	alces	R	0:04	1	ip-10-75-1-152
46	all	simplejo	alces	R	0:04	1	ip-10-75-1-152
47	all	simplejo	alces	R	0:04	1	ip-10-75-1-163
48	all	simplejo	alces	R	0:04	1	ip-10-75-1-163

```

[alces@login1(scooby) ~]$ scancel 47
[alces@login1(scooby) ~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
43	all	simplejo	alces	R	0:11	1	ip-10-75-1-50
44	all	simplejo	alces	R	0:11	1	ip-10-75-1-50
45	all	simplejo	alces	R	0:11	1	ip-10-75-1-152
46	all	simplejo	alces	R	0:11	1	ip-10-75-1-152
48	all	simplejo	alces	R	0:11	1	ip-10-75-1-163

Viewing compute host status

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

```
[alces@login1(scooby) ~]$ sinfo -Nl
Fri Aug 26 14:46:34 2016
NODELIST          NODES PARTITION      STATE CPUS      S:C:T MEMORY TMP_DISK WEIGHT_
↪AVAIL_FE REASON
ip-10-75-1-50      1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
ip-10-75-1-152    1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
ip-10-75-1-163    1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
ip-10-75-1-203    1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
ip-10-75-1-208    1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
ip-10-75-1-240    1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
ip-10-75-1-246    1      all*      idle   2      2:1:1   3602    20462    1
↪(null) none
```

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

- The hostname of your compute nodes
- The number of nodes in the list
- The node partition the node belongs to
- Current usage of the node - if no jobs are running, the state will be listed as `idle`. If a job is running, the state will be listed as `allocated`
- The detected number of CPUs (including hyper-threaded cores)
- The number of sockets, cores and threads per node
- The amount of memory in MB per node
- The amount of disk space in MB available to the `/tmp` partition per node
- The scheduler weighting

Default resources

In order to promote efficient usage of your cluster, the job-scheduler automatically sets a number of default resources for 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
- Number of nodes for your job: the default behavior is to allocate enough nodes to satisfy the requirements of the number of CPUs requested

You can view all default resource limits by running the following command:

```
[root@login1(slurm) ~]# scontrol show config | grep Def
CpuFreqDef          = Unknown
DefMemPerNode       = UNLIMITED
MpiDefault          = none
SallocDefaultCommand = (null)
```

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.

Controlling resources

In order to promote efficient usage of the cluster - the job-scheduler is automatically configured with default run-time limits for jobs. These defaults can 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 the default limits are applied to a job.

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

1. **On the command line**, as parameters to your `sbatch` or `srun` command. For example, you can set the name of your job using the `--job-name=[name]` | `-J [name]` option:

```
[alces@login1(scooby) ~]$ sbatch --job-name=mytestjob simplejobscript.sh
Submitted batch job 51

[alces@login1(scooby) ~]$ squeue
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       51         all mytestjo    alces  R        0:02      1 ip-10-75-1-50
```

2. **In your job script**, by including scheduler directives at the top of your job script - you can achieve the same effect as providing options with the `sbatch` or `srun` commands. Create an example job script or modify your existing script to include a scheduler directive to use a specified job name:

```
#!/bin/bash -l
#SBATCH --job-name=mytestjob
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 directives must start with `#SBATCH` at the beginning of the line
- You can have multiple lines starting with `#SBATCH` in your job-script, with normal script lines in-between
- You can put multiple instructions separated by a space on a single line starting with `#SBATCH`
- 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.
- Instructions are parsed at job submission time, before the job itself has actually run. This 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
- `%a / $SLURM_ARRAY_TASK_ID` Job array ID (index) number. The `%a` substitution should only be used in your job scheduler directives
- `%A / $SLURM_ARRAY_JOB_ID` Job allocation number for an array job. The `%A` substitution should only be used in your job scheduler directives
- `%j / $SLURM_JOBID` Job allocation number. The `%j` substitution should only be used in your job scheduler directives

Simple scheduler instruction examples

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

Setting output file location

To set the output file location for your job, use the `-o [file_name] | --output=[file_name]` option - both standard-out and standard-error from your job-script, including any output generated by applications launched by your job-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 and be writable by your user **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.

The following example uses the `--output=[file_name]` instruction to set the output file location:

```
#!/bin/bash -l
#SBATCH --job-name=myjob --output=output.%j

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

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

- `-D | --workdir=[dir_name]` - instruct the job scheduler to move into the directory specified 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 `-d [state:job_id] | --depend=[state:job_id]` option. For example, to wait until the job with ID 75 has finished before starting the job, you could use the following syntax:

```
[alces@login1(scooby) ~]$ squeue
      JOBID PARTITION    NAME    USER ST       TIME  NODES NODELIST(REASON)
       75      all    myjob    alces  R        0:01      1 ip-10-75-1-50

[alces@login1(scooby) ~]$ sbatch --dependency=afterok:75 mytestjob.sh
Submitted batch job 76

[alces@login1(scooby) ~]$ squeue
      JOBID PARTITION    NAME    USER ST       TIME  NODES NODELIST(REASON)
       76      all    myjob    alces  PD        0:00      1 (Dependency)
       75      all    myjob    alces  R        0:15      1 ip-10-75-1-50
```

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 `-a [array_spec] | --array=[array_spec]` directive. Your job-script can then use the pseudo environment variables created by the scheduler to refer to data used by each task in the job. The following job-script uses the `$SLURM_ARRAY_TASK_ID/%a` variable to echo its current task ID to an output file:

```
#!/bin/bash -l
#SBATCH --job-name=array
#SBATCH -D /home/alces/
#SBATCH --output=output.array.%A.%a
#SBATCH --array=1-1000
echo "I am $SLURM_ARRAY_TASK_ID from job $SLURM_ARRAY_JOB_ID"
```

```
[alces@login1(scooby) ~]$ sbatch arrayjob.sh
Submitted batch job 77
[alces@login1(scooby) ~]$ squeue
      JOBID PARTITION    NAME    USER ST       TIME  NODES NODELIST(REASON)
 77_[85-1000]      all    array    alces  PD        0:00      1 (Resources)
      77_71      all    array    alces  R        0:00      1 ip-10-75-1-163
      77_72      all    array    alces  R        0:00      1 ip-10-75-1-240
      77_73      all    array    alces  R        0:00      1 ip-10-75-1-163
      77_74      all    array    alces  R        0:00      1 ip-10-75-1-240
      77_75      all    array    alces  R        0:00      1 ip-10-75-1-246
      77_76      all    array    alces  R        0:00      1 ip-10-75-1-246
      77_77      all    array    alces  R        0:00      1 ip-10-75-1-208
      77_78      all    array    alces  R        0:00      1 ip-10-75-1-208
      77_79      all    array    alces  R        0:00      1 ip-10-75-1-152
      77_80      all    array    alces  R        0:00      1 ip-10-75-1-203
      77_81      all    array    alces  R        0:00      1 ip-10-75-1-50
      77_82      all    array    alces  R        0:00      1 ip-10-75-1-50
      77_83      all    array    alces  R        0:00      1 ip-10-75-1-152
      77_84      all    array    alces  R        0:00      1 ip-10-75-1-203
```

All tasks in an array job are given a job ID with the format `[job_ID]_[task_number]` e.g. `77_81` would be job number 77, array task 81.

Array jobs can easily be cancelled using the `scancel` command - the following examples show various levels of control over an array job:

scancel 77 Cancels all array tasks under the job ID 77

scancel 77_[100-200] Cancels array tasks 100-200 under the job ID 77

scancel 77_5 Cancels array task 5 under the job ID 77

Requesting more resources

By default, jobs are constrained to the default set of resources - 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 specifying the `-n`, `--ntasks=<number>` option in either your submission command or the scheduler directives in your job script. The `--ntasks` option informs the scheduler of the number of cores you wish to reserve for use. If the parameter is omitted, the default `--ntasks=1` is assumed. You could specify the option `-n 4` to request 4 CPU cores for your job.

Note: If you request more cores than are available on a node in your cluster, the job will not run until a node capable of fulfilling your request becomes available. The scheduler will display the error in the output of the `squeue` command

Running Parallel (MPI) jobs

If users want to run parallel jobs via a messaging 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 specifying the `-N`, `--nodes=<minnodes[-maxnodes]>` option - which requests a minimum (and optional maximum) number of nodes to allocate to the submitted job. If *only* the `minnodes` count is specified - then this is used for both the minimum *and* maximum node count for the job.

You can request multiple cores over multiple nodes using a combination of scheduler directives either in your job submission command or within your job script. Some of the following examples demonstrate how you can obtain cores across different resources;

--nodes=2 --ntasks=16 Requests 16 cores across 2 compute nodes

--nodes=2 Requests all available cores of 2 compute nodes

--ntasks=16 Requests 16 cores across any available compute nodes

For example, to use 64 CPU cores on the cluster for a single application, the instruction `--ntasks=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 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**.

```
#!/bin/bash -l
#SBATCH -n 64
#SBATCH --job-name=imb
#SBATCH -D /home/alces/
#SBATCH --output=imb.out.%j
module load apps/imb
mpirun --prefix $MPI_HOME \
      IMB-MPI1
```

We can then submit the IMB job script to the scheduler, which will automatically determine which nodes to use:

```
[alces@login1(scooby) ~]$ sbatch imb.sh
Submitted batch job 1162
[alces@login1(scooby) ~]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      1162      all      imb     alces  R        0:01      8 ip-
→10-75-1-[42,45,62,67,105,178,233,250]
[alces@login1(scooby) ~]$ cat imb.out.1162
#-----
#      Intel (R) MPI Benchmarks 4.0, MPI-1 part
#-----
# Date                : Tue Aug 30 10:34:08 2016
# Machine              : x86_64
# System               : Linux
# Release              : 3.10.0-327.28.3.el7.x86_64
# Version              : #1 SMP Thu Aug 18 19:05:49 UTC 2016
# MPI Version          : 3.0
# MPI Thread Environment:
#-----
# Benchmarking PingPong
# #processes = 2
# ( 62 additional processes waiting in MPI_Barrier)
#-----
#bytes #repetitions      t[usec]    Mbytes/sec
      0           1000        3.17         0.00
      1           1000        3.20         0.30
      2           1000        3.18         0.60
      4           1000        3.19         1.19
      8           1000        3.26         2.34
     16           1000        3.22         4.74
     32           1000        3.22         9.47
     64           1000        3.21        19.04
    128           1000        3.22        37.92
    256           1000        3.30        73.90
    512           1000        3.41       143.15
   1024           1000        3.55       275.36
   2048           1000        3.75       521.04
   4096           1000       10.09       387.14
   8192           1000       11.12       702.51
  16384           1000       12.06      1296.04
  32768           1000       14.65      2133.32
  65536           640       19.30      3238.72
 131072           320       29.50      4236.83
 262144           160       48.17      5189.77
 524288            80       84.36      5926.88
1048576            40      157.40     6353.32
2097152            20     305.00     6557.31
```


4194304	10	675.20	5924.16
---------	----	--------	---------

Note: If you request more CPU cores than your cluster can accommodate, your job will wait in the queue. If you are using the Flight Compute auto-scaling feature, your job will start to run once enough new nodes have been launched.

Requesting more memory

In order to promote best use of the cluster scheduler - particularly in a shared environment, it is recommended to inform the scheduler the maximum required memory per submitted job. This helps the scheduler appropriately place jobs on the available nodes in the cluster.

You can specify the maximum amount of memory required per submitted job with the `--mem=<MB>` option. This informs the scheduler of the memory required for the submitted job. Optionally - you can also request an amount of memory *per CPU core* rather than a total amount of memory required per job. To specify an amount of memory to allocate *per core*, use the `--mem-per-cpu=<MB>` option.

Note: When running a job across multiple compute hosts, the `--mem=<MB>` option informs the scheduler of the required memory *per node*

Requesting a longer runtime

In order to promote best-use of the cluster scheduler, particularly in a shared environment, it is recommend to inform the scheduler the amount of time the submitted job is expected to take. You can inform the cluster scheduler of the expected runtime using the `-t`, `--time=<time>` option. For example - to submit a job that runs for 2 hours, the following example job script could be used:

```
#!/bin/bash -l
#SBATCH --job-name=sleep
#SBATCH -D /home/alces/
#SBATCH --time=0-2:00
sleep 7200
```

You can then see any time limits assigned to running jobs using the command `squeue --long`:

```
[alces@login1(scooby) ~]$ squeue --long
Tue Aug 30 10:55:55 2016
      JOBID PARTITION     NAME     USER      STATE      TIME  TIME_LIMI  NODES
↳NODELIST (REASON)
      1163         all      sleep     alces    RUNNING      0:07   2:00:00       1
↳ip-10-75-1-42
```

Further documentation

This guide is a quick overview of some of the many available options of the SLURM cluster scheduler. For more information on the available options, you may wish to reference some of the following available documentation for the demonstrated SLURM commands;

- Use the `man squeue` command to see a full list of scheduler queue instructions
- Use the `man sbatch/srun` command to see a full list of scheduler submission instructions

- Online documentation for the SLURM scheduler is [available here](#)

OpenLava Scheduler

The [OpenLava](#) cluster job-scheduler is an open-source IBM Platform LSF compatible job-scheduler. The syntax and usage of commands is identical to the IBM Platform LSF scheduler, and users can typically migrate from one to another with no issues. This documentation provides a guide to using OpenLava 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 following example command:

```
bsub -Is bash
```

The above command uses the following options in order to create an interactive job, with shell mode support:

-Is Submits an interactive job and creates a pseudo-terminal when the job starts

bash Choose the shell type to use when creating the interactive job

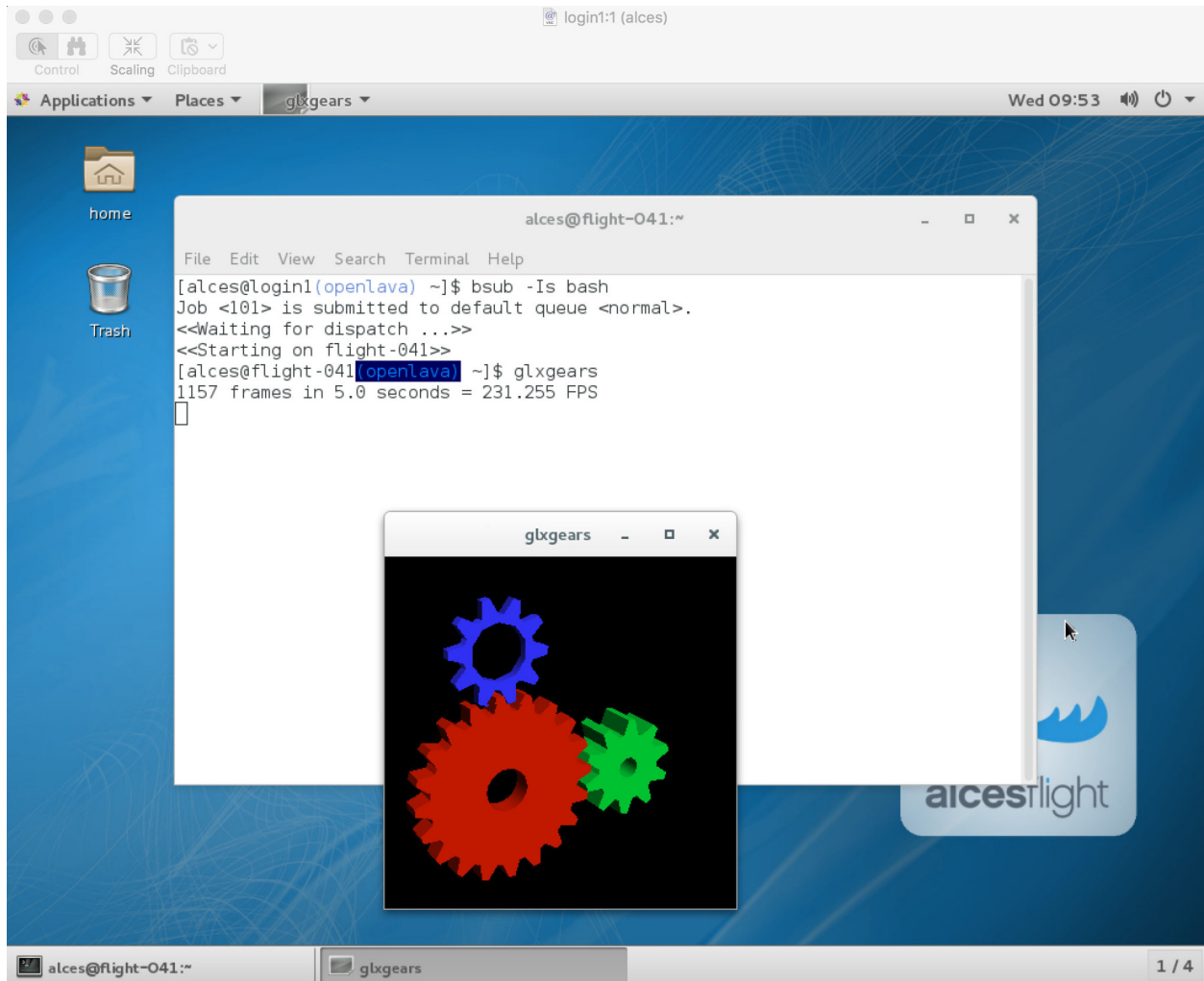
```
[alces@login1(scooby) ~]$ bsub -Is bash
Job <104> is submitted to default queue <normal>.
<<Waiting for dispatch ...>>
<<Starting on ip-10-75-1-61>>

[alces@ip-10-75-1-61(scooby) ~]$ module load apps/R
Installing distro dependency (1/1): tk ... OK

[alces@ip-10-75-1-61(scooby) ~]$ R

R version 3.3.0 (2016-05-03) -- "Supposedly Educational"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
```

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



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

Submitting a batch job

Batch (or non-interactive) jobs allow users to leverage one of the main benefits of a cluster scheduler; jobs can be queued up with instructions on how to run them and 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. This 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 long session. This ensures that environment modules can be loaded as required as part of your script.

Note: By default, if no output directory is specified - OpenLava will write all output files to the directory that the job script was submitted from. For example, if you submit the job script from the `/home/alces/job-scripts/example/` directory - all output files will attempt (if the compute hosts have access to that directory) to write all output files to the `/home/alces/job-scripts/example/` directory.

Warning: When submitting a batch job script through the `bsub` command - your job script should be redirected through `stdin`, for example `bsub < myscript.sh`. Failing to do so will cause your job to fail.

We can execute the job-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 `bsub < 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) ~]$ bsub < simplejobscript.sh
Job <151> is submitted to default queue <normal>.
```

Viewing and controlling queued jobs

Once your job has been submitted, use the `bjobs` command to see where they run. view the status of the job queue. If you have available compute nodes, your job should be shown in `RUN` (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 `PEND` (pending) state until compute nodes are available to run it.

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 `bkill <job-ID>` command to delete a job you've submitted, whether it's running or still in queued state.

```
[alces@login1(scooby) ~]$ bsub < simplejobscript.sh
Job <164> is submitted to default queue <normal>.
[alces@login1(scooby) ~]$ bsub < simplejobscript.sh
Job <165> is submitted to default queue <normal>.
[alces@login1(scooby) ~]$ bsub < simplejobscript.sh
Job <166> is submitted to default queue <normal>.
[alces@login1(scooby) ~]$ bkill 165
Job <165> is being terminated
[alces@login1(scooby) ~]$ bjobs
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
162	alces	RUN	normal	login1	ip-10-75-1-	sleep	Aug 30 16:15
163	alces	RUN	normal	login1	ip-10-75-1-	sleep	Aug 30 16:15
164	alces	PEND	normal	login1		sleep	Aug 30 16:15
166	alces	PEND	normal	login1		sleep	Aug 30 16:15

Viewing compute host status

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

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

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
ip-10-75-1-57	ok	-	2	0	0	0	0	0
ip-10-75-1-58	ok	-	2	0	0	0	0	0
ip-10-75-1-6	ok	-	2	0	0	0	0	0
ip-10-75-1-61	ok	-	2	0	0	0	0	0
ip-10-75-1-68	closed	-	2	2	2	0	0	0
login1	closed	-	0	0	0	0	0	0

The `bhosts` output shows information about the jobs running on each cluster scheduler host. You may also use the `-l` option to display more detailed information about each cluster execution host.

Default resources

By default, the OpenLava scheduler sets the default resource limits to “unlimited” if the resource is not specified in your job submission script or command. To promote efficient usage of the cluster scheduler, it is recommended to make use of the scheduler submission directives, which allow you to inform the scheduler how much of each resource a job may require. Informing the scheduler of the required resources will help you to better schedule and backfill jobs. The sections below detail how to inform the scheduler how much of various resource your job may require.

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 `bsub` command

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

```
[alces@login1(scooby) ~]$ bsub -J sleep < simplejobscript.sh
Job <167> is submitted to default queue <normal>.
```

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

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
167	alces	PEND	normal	login1		sleep	Aug 30 16:36

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

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

```
#!/bin/bash -l
#BSUB -J job_name
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 `#BSUB` at the beginning of the line
- You can have multiple lines starting with `#BSUB` in your job-script, with normal script lines in-between.
- You can put multiple instructions separated by a space on a single line starting with `#BSUB`
- 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
- Instructions provided as parameters to the `bsub` command 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
- `$HOSTNAME` The Linux hostname of the compute node running the job
- `$LSF_JOBID` The job-ID number for the job
- `$LSB_JOBINDEX` 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. If no job output directory is specified in your scheduler directives, the files will attempt to write to the directory the job script was submitted from.

By default, the scheduler stores data relative to the job submission 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 and be writeable **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
#BSUB -o /home/alces/outputs/test_jobs/sleep.$LSF_JOBID.out
echo "Hello from $HOSTNAME"
sleep 60
echo "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/test_jobs/sleep.24.out`.

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 `-w <dependency_expression>` 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 job that will only start running once job number 102 has finished, use the following example submission command:

```
[alces@login1(scooby) ~]$ bsub -w "done(101)" < myjobscript.sh
```

The job will then stay in pending status until the specified job number has reached completion. You can check the dependency exists by running the following command, which shows more detailed information about a job:

```
[alces@login1(scooby) ~]$ bjobs -l <job-ID>
Job Id <102>, User <alces>, Project <default>, Status <PEND>, Queue <normal>, Command
↳<#!/bin/bash -l;sleep 120>
Wed Aug 31 11:33:42: Submitted from host <login1>, CWD <$HOME>, Dependency Condition
↳<done(101)>;
PENDING REASONS:
Job dependency condition not satisfied: 1 host;
```

You can also depend on multiple jobs finishing before running a job - using the following example command;

```
[alces@login1(scooby) ~]$ bsub -w "done(103) && done(104)" < myjobscript.sh
Job <105> is submitted to default queue <normal>.

[alces@login1(scooby) ~]$ bjobs -l 105

Job Id <105>, User <alces>, Project <default>, Status <PEND>, Queue <normal>, Command
↳<#!/bin/bash -l;sleep 120>
Wed Aug 31 11:45:27: Submitted from host <login1>, CWD <$HOME>, Dependency Condition
↳<done(103) && done(104)>;
PENDING REASONS:
Job dependency condition not satisfied: 1 host;
```

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 `bsub` command together with the appropriate array syntax `-J name[array_spec]` in your job name. Your job-script can then use pseudo environment variables created by the scheduler to refer to data used by each task in the job. For example, the following job-script uses the `$LSF_JOBINDEX` variable to echo its current task ID to an output file. The job script also uses the scheduler directive `-o <output>` to specify an output file location. Using the variable substitutions `%J` and `%I` in the output specification allows the scheduler to generate a dynamic filename based on the job ID (`%J`) and array job index (`%I`) - generating the example output file `/home/alces/outputs/array/output.24.2` for job ID 24, array task 2.

```
#!/bin/bash -l
#BSUB -o /home/alces/outputs/array/output.%J.%I
echo "I am $LSB_JOBINDEX"
```

You can submit an array job using the syntax `-J "jobname[array_spec]"` - for example to submit an array job with the name `array` and 20 consecutively numbered tasks - you could use the following job submission line together with the above example jobscript:

```
bsub -J "array[1-20]" < array_job.sh
```

By including the following line, a separate output file for each task of the array job, for example task 22 of job ID 77 would generate the output file `output.74.22` in the specified directory.

```
#BSUB -o /home/alces/outputs/array/output.%J-%I
```

Array jobs can easily be cancelled using the `bkill` command - the following example shows various levels of control over an array job:

bkill 77 Cancels all array tasks under the job ID 77

bkill "77[1-100]" Cancels array tasks 1-100 under the job ID 77

bkill "77[22]" Cancels array task 22 under the job ID 77

Requesting more resources

By default, jobs are constrained to the default set of resources - 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 specifying the `-n <number of cores>` option in either your submission command or the scheduler directives in your job script. The `-n` option informs the scheduler of the number of cores you wish to reserve for use. For example; you could specify the option `-n 4` to request 4 CPU cores for your job.

Note: If the number of cores specified is more than the total amount of cores available on the cluster, the job will refuse to run and display an error

Running Parallel (MPI) jobs

If users want to run parallel jobs via a 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 specifying the `-n <number of cores>` option in either your submission

command or the scheduler directives in your job script. If the number of cores requested is more than any single node in your cluster, the job will be appropriately placed over two or more compute hosts as required.

For example, to use 64 cores on the cluster for a single application - the instruction `-n 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 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**. Using the scheduler directive `-R "span[ptile=8]"` allows you span each of the requested cores in the `-n 64` directive over as many nodes as are required, for example `-n 64 -R "span[ptile=8]"` would spread the job over 8 nodes, using 8 cores across each node - totaling 64 nodes.

```
#!/bin/bash -l
#BSUB -n 64 # Define the total number of cores to use
#BSUB -R "span[ptile=8]" # Number of cores per node
#BSUB -o imb.%J # Set output file to imb.<job-ID>
#BSUB -J mpi_imb # Set job name
module load apps/imb # Load required modules
machinefile=/tmp/machines.$$
for host in $LSB_HOSTS; do # generate node list
    echo $host >> $machinefile
done
mpirun --prefix $MPI_HOME \
    --hostfile $machinefile \
    $(which IMB-MPI1) PingPong # run IMB
rm -fv $machinefile # remove node list
```

The job script requests a total of 64 cores, requesting 8 cores on each compute host. The `-R "span[ptile=8]"` option can be used to specify the number of cores required per compute host.

Warning: Users running OpenJava may need to explicitly provide the number of MPI processes you wish to spawn as an option to the `mpirun` command. For example, to run 64 processes, the command `mpirun -np 64` would be used. The above example job script demonstrates several additionally required options in the `mpirun` command - most importantly `-np <number>` and `-npnnode <number>`. These options define the total number of MPI processes, as well as the number of MPI processes per node to spawn.

Note: If the number of cores specified is more than the total amount of cores available on the cluster, the job will not be scheduled to run and will display an error.

Requesting more memory

In order to promote best-use of the cluster scheduler - particularly in a shared environment, it is recommended to inform the scheduler the maximum required memory per submitted job. This helps the scheduler appropriately place jobs on the available nodes in the cluster.

You can specify the maximum amount of memory required per submitted job with the `-M [KB]` option. This informs the scheduler of the memory required for the submitted job. The following example job script can be used to submit a job which informs the scheduler your job may use up to 512MB of memory:

```
#!/bin/bash
#BSUB -o sleep.%J
#BSUB -M 512000
```

Requesting a longer runtime

In order to promote best-use of the cluster scheduler, particularly in a shared environment, it is recommended to inform the scheduler the amount of time the submitted job is expected to take. You can inform the cluster scheduler of the expected runtime using the `-W [hh:mm:ss]` option. For example - to submit a job that runs for 2 hours, the following example job script could be used:

```
#!/bin/bash -l
#BSUB -J sleep
#BSUB -o sleep.%J
#BSUB -W 02:00:00
```

Users can view any time limits assigned to running jobs using the command `bjobs -l [job-ID]`:

```
Job Id <117>, User <alces>, Project <default>, Status <RUN>, Queue <normal>, Command <
↪#!/bin/bash -l;sleep 120>
Wed Aug 31 13:31:18: Submitted from host <login1>, CWD <$HOME>;

RUNLIMIT
120.0 min of ip-10-75-1-
Wed Aug 31 13:31:25: Started on <ip-10-75-1-96>, Execution Home </home/alces>, ↪
↪Execution CWD </home/alces>;
Wed Aug 31 13:31:39: Resource usage collected.
                        MEM: 5 Mbytes;  SWAP: 346 Mbytes
                        PGID: 27789;   PIDs: 27789 27791 27794 2785
```

Further documentation

This guide is a quick overview of some of the many available options of the OpenLava cluster scheduler. For more information on the available options, you may wish to reference some of the following available documentation for the demonstrated OpenLava commands;

- Use the `man bjobs` command to see a full list of scheduler queue instructions
- Use the `man bsub` command to see a full list of scheduler submission instructions
- Online documentation for the OpenLava scheduler is [available here](#)

TORQUE Scheduler

The [Torque](#) cluster job scheduler is an open-source scheduler based on the original PBS codebase. TORQUE can be integrated with both the open-source Maui cluster scheduler or the commercial Moab workload manager.

See [Cluster job schedulers](#) for a description of the different use-cases of a cluster job-scheduler.

Running an interactive job

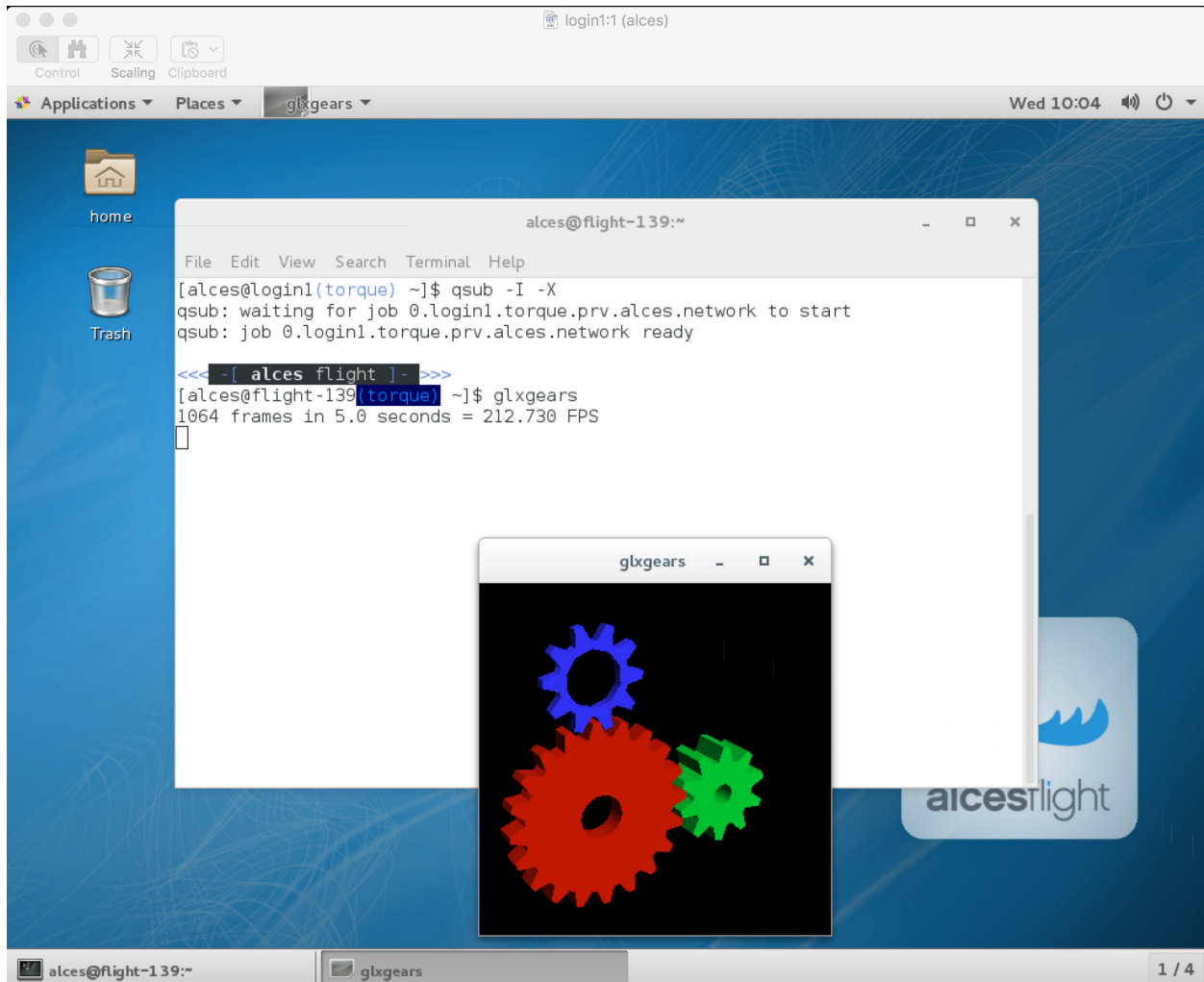
You can start a new interactive job on your Flight Compute cluster by using the `qsub -I` command; the scheduler will search for an available compute node, and provide you with an interactive login shell on the node if one is available.

```
[alces@login1(scooby) ~]$ qsub -I
qsub: waiting for job 2.login1.scooby.prv.alces.network to start
qsub: job 2.login1.scooby.prv.alces.network ready
```

```
<<< -[ alces flight ]- >>>
[alces@node-xb3(scooby) ~]$ hostname -f
node-xb3.scooby.prv.alces.network
```

In the above command, the `qsub` command is used together with the option `-I` which informs the cluster scheduler you wish to start an interactive job.

Alternatively, the `qsub -I` 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 interactive session are executed on the assigned cluster compute node.



Note: In order to run graphical applications within an interactive session, you must launch your interactive session with the `-X` option, which enables X forwarding. Launch an interactive session with X forwarding with the following command: `qsub -I -X`

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

If the job-scheduler could not satisfy the resource you've requested for your interactive job (e.g. all your available compute nodes are busy running other jobs), the job will queue until resources are available:

```
[alces@login1(scooby) ~]$ qsub -I
qsub: waiting for job 5.login1.scooby.prv.alces.network to start
```

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:

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

To submit your job script 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
7.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ cat simplejobscript.sh.o7
Running on host node-xb3
Finished running - goodbye from node-xb3
```

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 the `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 the `Q` (queued) state until compute nodes are available to run it. Jobs shown in `C` state have completed, and are automatically removed from the job queue after a few minutes.

You can keep running the `qstat` command until your job finishes running. 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. You can use the Linux `more` command to view your output file:

```
[alces@login1(scooby) ~]$ more simplejobscript.sh.o26
Running on host node-x4a
Finished running - goodbye from node-x4a
```

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 -n` command, see which node each job is running on.

```
[alces@login1(scooby) ~]$ qstat -n

login1.scooby.prv.alces.network:
Req'd      Req'd      Elap
Job ID      Time      Username  Queue   Jobname          SessID  NDS   TSK   _
--Memory    S      Time
-----
12.login1.scooby.prv.alce  alces      batch    simplejobscript.  7320     1     1   _
--      01:00:00 R  00:01:46
node-x4a
13.login1.scooby.prv.alce  alces      batch    simplejobscript.  9602     1     1   _
--      01:00:00 R  00:01:48
node-xb3
14.login1.scooby.prv.alce  alces      batch    simplejobscript.  4286     1     1   _
--      01:00:00 R  00:01:49
node-xd2
```

The scheduler is likely to spread jobs around over different nodes (if you have multiple nodes). The login node is not included in your cluster for scheduling purposes - jobs submitted to the scheduler will only 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 the queued state.

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
45.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
46.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
47.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
48.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qdel 47
[alces@login1(scooby) ~]$ qstat
```

Job ID	Name	User	Time Use	S	Queue
45.login1	...ejobscript.sh	alces	0	R	batch
46.login1	...ejobscript.sh	alces	0	R	batch
47.login1	...ejobscript.sh	alces	00:00:00	C	batch
48.login1	...ejobscript.sh	alces	0	R	batch

Viewing compute host status

Users can use the `pbsnodes -a` or `pbsnodes -l 'up'` options to view cluster node information. Any options other than `-l` or `-a` require PBS manager or PBS operator privileges.

Users can view compute host status in the following formats:

```

[root@login1(scooby) ~]# pbsnodes -l 'up'
node-xb3.scooby.prv.alc free
node-x4a.scooby.prv.alc free
node-xd2.scooby.prv.alc free
node-x94.scooby.prv.alc free
[root@login1(scooby) ~]# pbsnodes -a
  node-xb3.scooby.prv.alces.network
    state = free
    power_state = Running
    np = 2
    ntype = cluster
    status = rectime=1473089112,macaddr=0a:d7:ca:29:2a:a7,cpuclock=Fixed,varattr=,
↪ jobs=, state=free,netload=123268589,gres=, loadave=0.00,ncpus=2,physmem=3689160kb,
↪ availmem=3390616kb,totmem=3689160kb,idletime=3992,nusers=0,nsessions=0,uname=Linux_
↪ node-xb3 3.10.0-327.18.2.el7.x86_64 #1 SMP Thu May 12 11:03:55 UTC 2016 x86_64,
↪ opsys=linux
    mom_service_port = 15002
    mom_manager_port = 15003

  node-x4a.scooby.prv.alces.network
    state = free
    power_state = Running
    np = 2
    ntype = cluster
    status = rectime=1473089112,macaddr=0a:fd:8b:97:43:f1,cpuclock=Fixed,varattr=,
↪ jobs=, state=free,netload=121838538,gres=, loadave=0.00,ncpus=2,physmem=3689160kb,
↪ availmem=3402548kb,totmem=3689160kb,idletime=2652,nusers=0,nsessions=0,uname=Linux_
↪ node-x4a 3.10.0-327.18.2.el7.x86_64 #1 SMP Thu May 12 11:03:55 UTC 2016 x86_64,
↪ opsys=linux
    mom_service_port = 15002
    mom_manager_port = 15003

  node-xd2.scooby.prv.alces.network
    state = free
    power_state = Running
    np = 2
    ntype = cluster
    status = rectime=1473089113,macaddr=0a:77:b2:48:26:93,cpuclock=Fixed,varattr=,
↪ jobs=, state=free,netload=119609907,gres=, loadave=0.00,ncpus=2,physmem=3689160kb,
↪ availmem=3402008kb,totmem=3689160kb,idletime=1443,nusers=0,nsessions=0,uname=Linux_
↪ node-xd2 3.10.0-327.18.2.el7.x86_64 #1 SMP Thu May 12 11:03:55 UTC 2016 x86_64,
↪ opsys=linux
    mom_service_port = 15002
    mom_manager_port = 15003

  node-x94.scooby.prv.alces.network
    state = free
    power_state = Running
    np = 2
    ntype = cluster
    status = rectime=1473089103,macaddr=0a:82:bd:7d:5d:dd,cpuclock=Fixed,varattr=,
↪ jobs=, state=free,netload=118696570,gres=, loadave=0.00,ncpus=2,physmem=3689160kb,
↪ availmem=3403592kb,totmem=3689160kb,idletime=1026,nusers=0,nsessions=0,uname=Linux_
↪ node-x94 3.10.0-327.18.2.el7.x86_64 #1 SMP Thu May 12 11:03:55 UTC 2016 x86_64,
↪ opsys=linux
    mom_service_port = 15002
    mom_manager_port = 15003

```

The `pbsnodes` output will display some of the following information about the compute hosts in your cluster:

- The hostname of your compute nodes
- The number of nodes in the list
- Current usage of the node - if no jobs are running, the state will be listed as `free`
- The detected number of CPUs (including hyper-threaded cores)
- The amount of memory in KB per node
- The amount of disk space available per node

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. If there is no default limit in place, the limit will be unlimited or not defined - it is important to inform the cluster scheduler how much of each resource you require.

- Maximum job runtime (in hours): 1
- Default number of nodes: 1

You can view any default limits in place on the default batch queue with the following command:

```
[root@login1(torque) ~]# qmgr -c 'list queue batch'
Queue batch
    queue_type = Execution
    total_jobs = 1
    state_count = Transit:0 Queued:0 Held:0 Waiting:0 Running:1 Exiting:0
↪Complete:0
    resources_default.nodes = 1
    resources_default.walltime = 01:00:00
    mtime = Mon Sep 19 09:18:33 2016
    resources_assigned.nodect = 1
    enabled = True
    started = True
```

Providing job-scheduler instructions

Users can help the scheduler to understand how you want it to run your job by providing instructions - job instructions can be provided in two ways; they are:

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

1. **On the command line**, as parameters to your `qsub` command. For example, you can set the name of your job using the `-N <name>` option:

```
[alces@login1(scooby) ~]$ qsub -N mytestjob simplejobscript.sh
49.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qstat
```

Job ID	Name	User	Time Use	S	Queue
49.login1	mytestjob	alces	0	R	batch

2. **In your job script**, by including the scheduler directives at the top of your job script - you can achieve the same effect as providing options with the `qsub` command. Lines in your script containing scheduler directives must start with `#PBS` and be located at the top of your script, after the shell line. Create an example job script or modify your existing script to include a scheduler directive to use a specified job name:

```
[alces@login1(scooby) ~]$ cat simplejobscript.sh
#!/bin/bash -l
#PBS -N mytestjob
echo "Running on host $HOSTNAME"
sleep 120
echo "Finished running - goodbye from $HOSTNAME"
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
51.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qstat
```

Job ID	Name	User	Time Use	S	Queue
49.login1	mytestjob	alces	00:00:00	C	batch
50.login1	mytestjob	alces		0	R batch
51.login1	mytestjob	alces		0	R batch

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 directives must start with `#PBS` at the beginning of the line
- You can have multiple lines starting with `#PBS` in your job-script, but they must appear at the top of the script without any lines in-between
- You can put multiple instructions separated by a space on a single line starting with `#PBS`
- 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
- Instructions are parsed at job submission time, before the job itself has actually run. This 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
- `$HOSTNAME` The Linux hostname of the compute node running the job
- `$PBS_JOBID` Job allocation number
- `$PBS_ARRAYID` Job array ID (index) number

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 [file_name]` option. This will send all `stdout` to the specified file. The `-e [file_name]` option can also be used to specify an output file for all `stderr`. If you wish to combine both `stdout` and `stderr` to the same output file - you can use the option `-j oe [file_name]`.

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(s) 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 [file_name]` instruction to set the output file location:

```
#!/bin/bash -l
#PBS -N mytestjob -o testjob.$PBS_JOBID
echo "Starting running on host $HOSTNAME"
sleep 120
echo "Finished running - goodbye from $HOSTNAME"
```

In the above example, assuming the job was submitted as the `alces` user and was given the job-ID number 53, the scheduler will save the output data from the job in the filename `/home/alces/testjob.52.login1.<clustername>.prv.alces.network`.

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

Setting working directory for your job

Torque uses the directory that the job was submitted from to define the working directory for a job - no matter the location of the job submission script. For example, on your cluster if you create a new directory in your home directory named `outputs` then `cd` to the `outputs` folder:

```
[alces@login1(scooby) ~]$ mkdir outputs && cd outputs
[alces@login1(scooby) outputs]$ pwd
/home/alces/outputs
```

You can then submit a job script that exists in any directory, and the job output and working directory will be the current working directory. The dynamic variable `$PBS_O_WORKDIR` variable should be used to determine the working directory. The following example job script demonstrates this functionality:

```
[alces@login1(scooby) outputs]$ cat ../wd.sh
#!/bin/bash -l
echo "My working directory is $PBS_O_WORKDIR"

[alces@login1(scooby) outputs]$ qsub ../wd.sh
30.login1.scooby.prv.alces.network

[alces@login1(scooby) outputs]$ cat wd.sh.o30
My working directory is /home/alces/outputs
```

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 `-W depend=[spec]` option. For example, to wait until the job ID 55 has finished, the following example command can be used:

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
55.login1.scooby.prv.alces.network

[alces@login1(scooby) ~]$ qsub -W depend=afterok:55 simplejobscript.sh
56.login1.scooby.prv.alces.network

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

Job ID	Name	User	Time Use	S	Queue
54.login1	mytestjob	alces	00:00:00	C	batch
55.login1	mytestjob	alces		0 R	batch
56.login1	mytestjob	alces		0 H	batch

Your job will be held in H (*hold*) state until the dependency condition is met.

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 [array_spec]` directive. Your job-script can then use the pseudo environment variables created by the scheduler to refer to data used by each task in the job. The following example job-script uses the `$PBS_ARRAYID` variable to echo its current task ID to an output file:

```
#!/bin/bash -l
#PBS -N array_job
#PBS -j oe array_job.$PBS_JOBID.$PBS_ARRAYID
#PBS -t 1-5
echo "Hello from $PBS_ARRAYID - part of $PBS_JOBID"
```

The example script will create output files for each of the task array jobs run through the scheduler:

```
[alces@login1(scooby) ~]$ ls
array_job.o59-1 array_job.o59-3 array_job.o59-5 clusterware-setup-sshkey.log
array_job.o59-2 array_job.o59-4 array_job.sh
[alces@login1(scooby) ~]$ cat array_job.o59-2
Hello from 2 - part of 59[2].login1.scooby.prv.alces.network
```

All tasks in an array job are given a job ID with the format `job_ID[task_number]`, e.g. `54[2]` would be job number 54, array task 2.

Array jobs can easily be cancelled using the `qdel` command - the following examples show various levels of control over an array job:

qdel 60[] Cancels all array tasks under the job ID 60

qdel -t 100-200 60[] Cancels array tasks 100-200 under the job ID 60

qdel -t 5 60[] Cancels array task 5 under the job ID 60

Note: When cancelling array tasks under an array job, the job ID number must include the two empty brackets [] as shown after the job ID

Requesting more resources

By default, jobs are constrained to a default set of resources - 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 specifying `-l mppwidth=[count]` option in either your submission command or the scheduler directives in your job script. The `-l mppwidth=[count]` option informs the scheduler of the number of cores you wish to reserve for use. If the parameter is omitted, a default of 1 core is assumed. You could specify the option `-l mppwidth=4` to request 4 CPU cores for your job.

Running Parallel (MPI) jobs

If users want to run parallel jobs via a 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 specifying the `-l nodes=X:ppn=Y` option either in your job submission command or your job-script directives, to request **Y** cores on each of **X** nodes.

For example, to use 8 CPU cores on the cluster for a single application - you could use the following scheduler directive:

```
-l nodes=4:ppn=2 Request 4 nodes using 2 cores across each requested node
```

The following example shows launching the **Intel Message-passing (IMB)** 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 specified as parameters to `mpirun`. This jobscript loads the `apps/imb` module before launching the application, which automatically loads the module for **OpenMPI**.

```
#!/bin/bash -l
#PBS -l nodes=8:ppn=8
#PBS -N imb
#PBS -j oe /home/alces/outputs/imb_mpi.out.$PBS_JOBID
module load apps/imb
echo "List of nodes to use:"
echo "-----"
cat $PBS_NODEFILE
mpirun --prefix $MPI_HOME \
      -np 8 \
      -npnode 2 \
      --hostfile $PBS_NODEFILE \
      $(which IMB-MPI1)
```

The above example job script demonstrates several additionally required options in the `mpirun` command - most importantly `-np <number>` and `-npnode <number>`. These options define the total number of MPI processes, as well as the number of MPI processes per node to spawn.

Once the above job-script is submitted to the job-scheduler, the required number of nodes will be allocated for execution of the workload; e.g.

```
[alces@login1(scooby) outputs]$ qsub ../imb_mpi.sh
35.login1.scooby.prv.alces.network

[alces@login1(scooby) outputs]$ cat imb.o35
List of nodes to use:
-----
node-x90.scooby.prv.alces.network
node-x90.scooby.prv.alces.network
node-xd7.scooby.prv.alces.network
node-xd7.scooby.prv.alces.network
node-x81.scooby.prv.alces.network
node-x81.scooby.prv.alces.network
node-xc3.scooby.prv.alces.network
node-xc3.scooby.prv.alces.network
benchmarks to run PingPong
#-----
#      Intel (R) MPI Benchmarks 4.0, MPI-1 part
#-----
# Date           : Tue Sep  6 10:26:04 2016
```

Note: If you request more CPU cores than your cluster can accommodate, your job will wait in the queue. If you are using the Flight Compute auto-scaling feature, your job will start to run once enough new nodes have been launched.

Requesting more memory

In order to promote best-use of the cluster scheduler - particularly in a shared environment, it is recommended to inform the scheduler the maximum required memory per submitted job. This helps the scheduler appropriately place jobs on the available nodes in the cluster.

You can specify the maximum amount of memory required per submitted job with the `-l mem=[XXXmb]` option. This informs the scheduler of the memory required for the submitted job. Optionally - you can also request an amount of memory *per CPU core* rather than a total amount of memory required per job.

Note: When running a job across multiple compute hosts, the `-l mem=[XXXmb]` option informs the scheduler of the required memory *per node*

Requesting a longer runtime

In order to promote best-use of the cluster scheduler, particularly in a shared environment, it is recommended to inform the scheduler of the amount of time the submitted job is expected to take. You can inform the cluster scheduler of the expected runtime using the `-l walltime=[hh:mm:ss]` option. For example - to submit a job that runs for a maximum of 2 hours, the following example job script could be used:

```
#!/bin/bash -l
#PBS -l walltime=02:00:00
```

Further documentation

This guide is a quick overview of some of the many available options of the TORQUE cluster scheduler. For more information on the available options, you may wish to reference some of the following available documentation for the demonstrated TORQUE commands;

- Use the `man qstat` command to see a full list of scheduler queue instructions
- Use the `man qsub` command to see a full list of scheduler submission instructions
- Online documentation for the TORQUE scheduler is [available here](#)

PBS Pro Scheduler

The **PBS Pro** cluster job scheduler is an open-source scheduler based on the original PBS codebase. PBS was originally developed by NASA - Altair Engineering now owns and maintains the PBS Pro cluster scheduler.

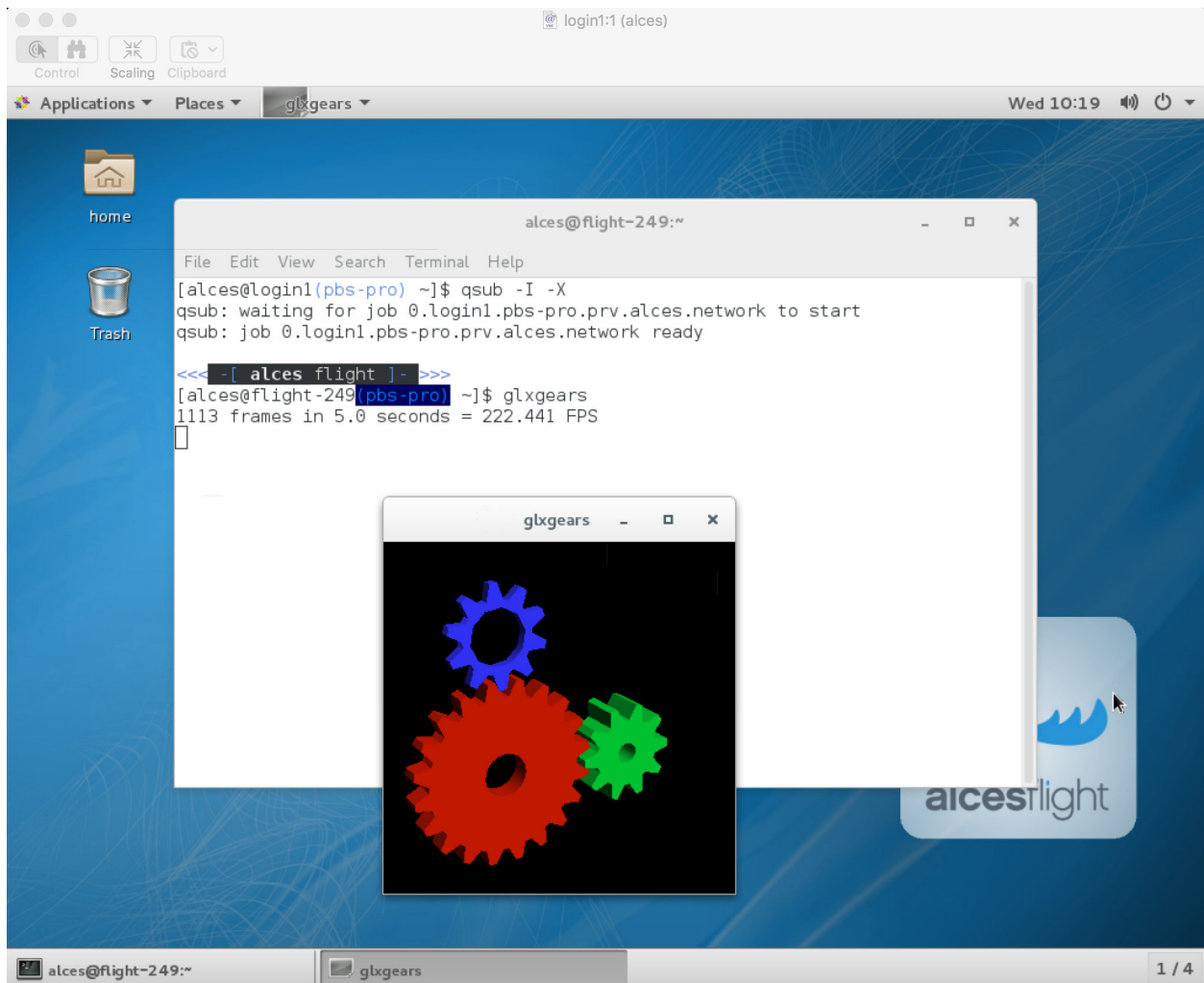
See *Cluster job schedulers* for a description of the different use-cases of a cluster job-scheduler.

Running an interactive job

You can start a new interactive job on your Flight Compute cluster by using the `qsub -I` 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.

In the above command, the `qsub` command is used together with the option `-I` which informs the cluster scheduler you wish to start an interactive job.

Alternatively, the `qsub -I` 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 interactive session are executed on the assigned cluster compute node.



Note: In order to run graphical applications within an interactive session, you must launch your interactive session with the `-X` option, which enables X forwarding. Launch an interactive session with X forwarding using the following command: `qsub -I -X`

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

If the job-scheduler could not satisfy the resource you've requested for your interactive job (e.g. all your available compute nodes are busy running other jobs), the job will queue until resources are available:

```
[alces@login1(scooby) ~]$ qsub -I
qsub: waiting for job 13.login1.scooby.prv.alces.network to start
```

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 `simplejobsscript.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:

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

To submit your job script 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
14.login1.scooby.prv.alces.network

[alces@login1(scooby) ~]$ cat simplejobscript.sh.o14
Starting running on host node-x3a
Finished running - goodbye from node-x3a
```

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 the `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 the `Q` (queued) state until compute nodes are available to run it. Jobs shown in `C` state have completed, and are automatically removed from the job queue after a few minutes.

You can keep running the `qstat` command until your job finishes running. 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. You can use the Linux `more` command to view your output file:

```
[alces@login1(scooby) ~]$ more simplejobscript.sh.o22
Starting running on host node-x3a
Finished running - goodbye from node-x3a
```

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 -n` command, see which node the job is running on.

```
[alces@login1(scooby) ~]$ qstat -n
```

login1.scooby.prv.alces.network:

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
23.login1.pbs-p node-x3a	alces	workq	simplejobs	5974	1	1	--	--	R	00:00
24.login1.pbs-p node-x3a	alces	workq	simplejobs	6068	1	1	--	--	R	00:00
25.login1.pbs-p node-x3a	alces	workq	simplejobs	6159	1	1	--	--	R	00:00

The scheduler is likely to spread jobs around over different nodes (if you have multiple nodes). The login node is not included in your cluster for scheduling purposes - jobs submitted to the scheduler will only 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 the queued state.

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
33.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
34.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
35.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qdel 34
[alces@login1(scooby) ~]$ qstat
```

Job id	Name	User	Time Use	S	Queue
33.login1	simplejobscript	alces	00:00:00	R	workq
35.login1	simplejobscript	alces	00:00:00	R	workq

Viewing compute host status

Users can use the `pbsnodes -av` command to display the currently active compute hosts and information associated to each of those nodes, for example:

```
[alces@login1(scooby) ~]$ pbsnodes -av
node-x3a.scooby.prv.alces.network
  Mom = node-x3a.scooby.prv.alces.network
  ntype = PBS
  state = free
  pcpus = 8
  resources_available.arch = linux
  resources_available.host = node-x3a
  resources_available.mem = 14973084kb
  resources_available.ncpus = 8
  resources_available.vnode = node-x3a.scooby.prv.alces.network
  resources_assigned.accelerator_memory = 0kb
  resources_assigned.mem = 0kb
  resources_assigned.naccelerators = 0
  resources_assigned.ncpus = 0
  resources_assigned.netwins = 0
  resources_assigned.vmem = 0kb
  resv_enable = True
  sharing = default_shared
  license = 1
```


The `pbsnodes` output will display the following information about the compute hosts in your cluster:

- The hostname of your compute nodes
- The number of nodes in the list
- Current usage of the node - if no jobs are running, the state will be listed as `free`
- The detected number of CPUs (including hyper-threaded cores)
- The amount of memory in KB per node
- The amount of disk space available per node

Controlling resources

In order to promote efficient usage of the cluster - the job-scheduler should be informed of any resource requirements of your submitted jobs in order to effectively fill the available compute hosts. By default, each of the available resource types are set to “unlimited”. Scheduling each of your jobs with “unlimited” definitions will mean it is harder to effectively schedule jobs. Defining the required resources means that the scheduler can better work out where a job should be placed to fully optimise the available compute estate.

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

1. **On the command line**, as parameters to your `qsub` command. For example, you can set the name of your job using the `-N <name>` option:

```
[alces@login1(scooby) ~]$ qsub -N mytestjob simplejobscript.sh
36.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qstat
```

Job id	Name	User	Time Use	S	Queue
36.login1	mytestjob	alces	00:00:00	R	workq

2. **In your job script**, by including the scheduler directives at the top of your job script - you can achieve the same effect as providing options with the `qsub` command. Lines in your script containing scheduler directives must start with `#PBS` and be located at the top of your script, after the shell line. Create an example job script or modify your existing script to include a scheduler directive to use a specified job name:

```
[alces@login1(scooby) ~]$ cat simplejobscript.sh
#!/bin/bash
#PBS -N mytestjob
echo "Starting running on host $HOSTNAME"
sleep 120
echo "Finished running - goodbye from $HOSTNAME"
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
37.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qstat
```

Job id	Name	User	Time Use	S	Queue
36.login1	mytestjob	alces	00:00:00	R	workq
37.login1	mytestjob	alces	00:00:00	R	workq

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 directives must start with `#PBS` at the beginning of the line
- You can have multiple lines starting with `#PBS` in your job-script, but they must appear at the top of the script without any lines in-between

- You can put multiple instructions separated by a space on a single line starting with #PBS
- 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
- Instructions are parsed at job submission time, before the job itself has actually run. This 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
- \$PBS_JOBID Job allocation number. If job is an array job, includes the array index
- \$PBS_ARRAY_INDEX Sub job index in job array, e.g. 7
- \$PBS_ARRAY_ID Identifier for a job array. Sequence number of job array, e.g. 1234 []

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 [file_name]` option. This will send all `stdout` to the specified file. The `-e [file_name]` option can also be used to specify an output file for all `stderr`. If you wish to combine both `stdout` and `stderr` to the same output file - you can use the option `-j oe`.

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(s) 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 [file_name]` instruction to set the output file location:

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

Note: PBS Pro does not support the use of dynamic environment variables within scheduler directives. You may use the `$PBS_JOBID` variable from within your job script, but not as part of the output file name

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

Setting working directory for your job

PBS Pro uses the directory that the job was submitted from to define the working directory for a job - no matter the location of the job submission script. For example, on your cluster if you create a new directory in your home directory named `outputs` then `cd` to the `outputs` folder:

```
[alces@login1(scooby) ~]$ mkdir outputs && cd outputs
[alces@login1(scooby) outputs]$ pwd
/home/alces/outputs
```

You can then submit a job script that exists in any directory, and the job output and working directory will be the current working directory. The dynamic variable `$PBS_O_WORKDIR` variable should be used to determine the working directory. The following example job script demonstrates this functionality:

```
[alces@login1(scooby) outputs]$ cat ../wd.sh
#!/bin/bash -l
echo "My working directory is $PBS_O_WORKDIR"
[alces@login1(scooby) outputs]$ qsub ../wd.sh
30.login1.scooby.prv.alces.network
[alces@login1(scooby) outputs]$ cat wd.sh.o30
My working directory is /home/alces/outputs
```

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 `-W depend=[spec]` option. For example, to wait until the job ID 55 has finished, the following example command can be used:

```
[alces@login1(scooby) ~]$ qsub simplejobscript.sh
55.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qsub -W depend=afterok:55 simplejobscript.sh
56.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qstat
```

Job ID	Name	User	Time Use	S	Queue
54.login1	mytestjob	alces	00:00:00	C	batch
55.login1	mytestjob	alces		0	R batch
56.login1	mytestjob	alces		0	H batch

Your job will be held in `H` (*hold*) state until the dependency condition is met.

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 `-J [array_spec]` directive. Your job-script can then use the pseudo environment variables created by the scheduler to refer to data used by each task in the job. The following example job-script uses the `$PBS_JOBID` variable to echo its current task ID to an output file:

```
#!/bin/bash -l
#PBS -N array_job
#PBS -J 1-5
echo "Hello from $PBS_JOBID - I am array task $PBS_ARRAY_INDEX"
```

The example script will create output files for each of the task array jobs run through the scheduler:

```
[alces@login1(scooby) ~]$ ls
array_job.o59-1  array_job.o59-3  array_job.o59-5  clusterware-setup-sshkey.log
array_job.o59-2  array_job.o59-4  array_job.sh
[alces@login1(scooby) ~]$ cat array_job.o59-2
Hello from 59[2].login1.scooby.prv.alces.network - I am array task 2
```

All tasks in an array job are given a job ID with the format `job_ID[task_number]`, e.g. `54[2]` would be job number 54, array task 2.

Array jobs can easily be cancelled using the `qdel` command - the following examples show various levels of control over an array job:

qdel 60 [] Cancels all array tasks under the job ID 60

qdel 60 [100-200] Cancels array tasks 100-200 under the job ID 60

qdel 60 [5] Cancels array task 5 under the job ID 60

Note: When cancelling array tasks under an array job, the job ID number must include the two empty brackets `[]` as shown after the job ID

Requesting more resources

By default, jobs are constrained to the default set of resources - 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 specifying `-l ncpus=[count]` option in either your submission command or as a scheduler directive in your job script. The `-l ncpus=[count]` option informs the scheduler of the number of cores you wish to reserve for use. If the parameter is omitted, a default of 1 core is assumed. For example, a user can specify the option `-l ncpus=4` to request 4 CPU cores for your job.

Running Parallel (MPI) jobs

If users want to run parallel jobs via a 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 specifying the following example option:

```
-l select=2:ncpus=1:mpiprocs=1 -l place=scatter
```

The above example would launch an MPI job with a total of 2 CPU cores across 2 separate compute hosts - each compute host would launch a single MPI task. The command consists of several options:

select=2 Select the number of *chunks* - a *chunk* is essentially a task, or MPI-rank

ncpus=1 Select the number of CPU cores to use per *chunk*

mpiprocs Select the number of MPI processes to launch per *chunk*. This should be equal to *ncpus*

place=scatter The *place* option determines where MPI processes will launch. If the *scatter* option is chosen - each *chunk* will be launched on a different compute host. Other available options are *free*, *pack* and *excl*.

The following example shows launching the **Intel Message-passing (IMB)** 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 specified as parameters to `mpirun`. This jobscript loads the `apps/imb` module before launching the application, which automatically loads the module for **OpenMPI**.

```
#!/bin/bash -l
#PBS -l select=4:ncpus=1:mpiprocs=1
#PBS -l place=scatter
#PBS -j oe
module load apps/imb
mpirun --prefix $MPI_HOME \
  -np 4 \
  -npernode 1 \
  --hostfile $PBS_NODEFILE \
  $(which IMB-MPI1)
```

The above example job script demonstrates several additionally required options in the `mpirun` command - most importantly `-np <number>` and `-npernode <number>`. These options define the total number of MPI processes, as well as the number of MPI processes per node to spawn.

Once the above job-script is submitted to the job-scheduler, the required number of nodes will be allocated for execution of the workload; e.g.

```
[alces@login1(scooby) ~]$ qsub imb_mpi.sh
77.login1.scooby.prv.alces.network
[alces@login1(scooby) ~]$ qstat -n

login1.scooby.prv.alces.network:


```

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
77.login1.pbs-p	alces	workq	imb_mpi.sh	14129	4	4	--	--	R	00:00
node-x3a+node-x4d+node-xe3+node-x70										

Note: If you request more CPU cores than your cluster can accommodate, your job will wait in the queue. If you are using the Flight Compute auto-scaling feature, your job will start to run once enough new nodes have been launched.

Requesting more memory

In order to promote best-use of the cluster scheduler - particularly in a shared environment, it is recommended to inform the scheduler the maximum required memory per submitted job. This helps the scheduler appropriately place jobs on the available nodes in the cluster.

You can specify the maximum amount of memory required per submitted job with the `-l mem=[XXXmb]` option. This informs the scheduler of the memory required for the submitted job.

Note: When running a job across multiple compute hosts, the `-l mem=[XXXmb]` option informs the scheduler of the required memory *per node*

Requesting a longer runtime

In order to promote best-use of the cluster scheduler, particularly in a shared environment, it is recommended to inform the scheduler of the amount of time the submitted job is expected to take. You can inform the cluster scheduler of the expected runtime using the `-l walltime=[hh:mm:ss]` option. For example - to submit a job that runs for a maximum of 2 hours, the following example job script could be used:

```
#!/bin/bash -l
#PBS -l walltime=02:00:00
sleep 120
```

Further documentation

This guide is a quick overview of some of the many available options of the PBS Pro cluster scheduler. For more information on the available options, you may wish to reference some of the following available documentation for the demonstrated PBS Pro commands;

- Use the `man qstat` command to see a full list of scheduler queue instructions
- Use the `man qsub` command to see a full list of scheduler submission instructions
- Online documentation for the PBS Pro scheduler is [available here](#)

Cluster customisation: AWS

An Alces Flight Compute environment contains many useful tool and utilities designed to assist users in running a diverse range of jobs and workflows. Also included are tools to help users customise their cluster environment further, allowing automated setup of a range of features including:

- Distribution package installation
- External storage mounts
- Custom script execution at launch time

The Alces customiser tool requires some setup tasks in order to appropriately work with your AWS account and deployed Alces Flight Compute environments. Follow the steps below to enable the customiser for your AWS account.

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; e.g.

```
[alces@login1(scooby) ~]$ alces about node
Clusterware release: 2016.3
Customiser bucket prefix: s3://alces-flight-ali0ytdmvzv3ztv3/customiser/default
Platform host name: ec2-52-51-77-141.eu-west-1.compute.amazonaws.com
Public IP address: 52.51.77.141
Account hash: ali0ytdmvzv3ztv3
```

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

```
alces-flight-ali0ytdmvzv3ztv3
```

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

```
customiser
```

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

From within the `default` folder, we must create a folder 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 that the Alces customiser will not be able to locate customisation information.

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

Using custom S3 buckets

You may also wish to use a custom S3 bucket rather than the automatically generated Flight bucket name. To do so, simply follow the above steps to create a bucket in the same location, changing `default` for a different identifier. For example, the following location could be created to hold customisation scripts for a specific environment:

```
s3://alces-flight-bluecluster/customiser/default
```

To use custom S3 buckets with the Alces Flight Compute 2016.3 release, enter your S3 bucket URL in the `FlightCustomBuckets` CloudFormation parameter, without the S3 prefix. For example, to launch a cluster using customisation scripts from the bucket in the above example, a user could specify the following value at launch time:

```
FlightCustomBuckets: alces-flight-ali0ytdmvzv3ztv3/customiser/
default
```

Setting up customisation scripts

Customisation scripts are run on each node in your environment upon joining the cluster network - example customisation scripts include distribution package installations and external storage mounts. The Alces customiser supports any Linux 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
```

Note: Customisation scripts are run in unattended mode, and should be written to complete with interactive input.

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>/customiser/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 the file `/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.

Using alternate customisation profiles

Alternate customisation profiles can be set up and used with the Alces customiser tool. To set up another profile, from your S3 bucket in the `customiser` folder - create another profile folder, for example `foo`

Within the `foo` folder - create the `configure.d` folder. Place any customisation scripts for the `foo` profile within the `configure.d` folder.

To use custom profiles when launching the Alces Flight Compute 2016.3 CloudFormation templates, enter the profile name(s) in the `FlightCustomProfiles` parameter - the customiser tool will then run each of the scripts in the `foo` profile.

Hyperthreading Control

Some Cloud instances are configured with “virtual” CPU cores. For each physical processor core, the OS addresses a second virtual core - this is known as [Hyperthreading](#). In some cases, your application may suffer a performance drop when Hyperthreading is enabled. Alces Flight Compute includes the necessary tools to easily and dynamically disable Hyperthreading on each of your instances.

Checking the status of Hyperthreading

You can check whether a node has Hyperthreading enabled or disabled by running the following command:

```
[alces@node-xf5(scooby) ~]$ alces configure hyperthreading
alces configure hyperthreading: hyperthreading is enabled
```

Note: Hyperthreading status should be checked, disabled or enabled on a per-node basis. Users can check all nodes in their cluster using `pdsh -e.g. pdsh -g cluster alces configure hyperthreading`.

Disabling Hyperthreading

Disabling Hyperthreading will dynamically turn off the secondary thread for each physical CPU core which your instance has access to. For most platforms, this will effectively halve the number of CPU cores reported by your instance. To see the current number of cores available to your instance, users can run the following example command:

```
[alces@node-xf5(scooby) ~]$ cat /proc/cpuinfo | grep processor | wc -l
8
```

Users can then disable Hyperthreading and confirm its status using the following commands:

```
[alces@node-xf5(scooby) ~]$ alces configure hyperthreading disable
alces configure hyperthreading: hyperthreading disabled
[alces@node-xf5(scooby) ~]$ cat /proc/cpuinfo | grep processor | wc -l
4
```

Note: Your cluster job-scheduler periodically checks compute nodes to confirm how many CPU cores are reported as online. After changing the hyperthreading setting for your compute nodes, your job-scheduler may take a few minutes to report the new online CPU count for each node.

Launching a single Alces Flight instance on AWS

Alces Flight Compute is provided as a single multi-purpose Amazon Machine Image (AMI) which can be configured to run as a cluster login or compute node on Amazon Web Services (AWS) public cloud. An Amazon CloudFormation template can be used to launch a number of instances at the same time, along with relevant infrastructure components (e.g. networks, security groups) to create a compute cluster quickly and easily. The AWS Marketplace provides users with a method to quickly locate the template and launch a cluster, based on a short list of configuration questions answered at launch time.

For advanced users, or users that only require a single login node for their work, Alces Flight Compute can also be launched as a single instance. Users have access to the full range of Gridware software applications available on a compute cluster and can choose to optionally add infrastructure and compute nodes to their environment at a later date.

Finding and Launching the Alces Flight AMI

To launch a single-instance of Alces Flight Compute use the AWS Marketplace to search for the latest version of Alces Flight Compute and subscribe to the product. Click on the “Custom Launch” tab and a list of AMI-IDs will be provided that instances can be launched from:

Launch on EC2:

Flight Compute cluster (Community Support)

1-Click Launch
Review, modify, and launch

Custom Launch
CloudFormation, EC2 Console, APIs or CLI

Launching Options

- You can click "Launch with CloudFormation Console" button below and follow the steps in the CloudFormation console to launch a stack of this software.
- If you prefer to launch just an AMI, you can select Single AMI and click the "Launch with EC2 Console" and follow the instructions to launch an instance of this software.
- If you want to only launch the AMI, you can also find and launch these AMIs by searching for the AMI IDs (shown below) in the "Community AMIs" tab of the [EC2 Console](#) Launch Wizard, or launch with the [EC2 APIs](#).
- You can view this information at a later time by visiting the Your Software page. For help, see [step-by-step instructions](#) for launching Marketplace Products from the AWS Console.

Select a Version

Select a Region

Deployment Options

- ☒ **Single AMI**
64-bit Amazon Machine Image (AMI) ([learn more](#))
Single box deployment of the product
- ☐ **Personal HPC compute cluster**
CloudFormation template ([view](#))
1 x on-demand login node plus a choice of compute nodes

Launch

AMI IDs

Region	ID	Launch with EC2 Console
US East (N. Virginia)	ami-db53a1b6	Launch with EC2 Console
US West (Oregon)	ami-9b5aa3fb	Launch with EC2 Console

Pricing Details

For Region
US East (N. Virginia)

Delivery Methods
Single AMI

Hourly Fees
Total hourly fees will vary by instance type and EC2 region.

EC2 Instance Type	Software	EC2	Total
t2.large	\$0.00/hr \$0.00/yr	\$0.104/hr	\$0.104/hr
m4.xlarge	\$0.00/hr \$0.00/yr	\$0.239/hr	\$0.239/hr
m4.2xlarge	\$0.00/hr \$0.00/yr	\$0.479/hr	\$0.479/hr
m4.4xlarge	\$0.00/hr \$0.00/yr	\$0.958/hr	\$0.958/hr
m4.10xlarge	\$0.00/hr \$0.00/yr	\$2.394/hr	\$2.394/hr
c4.large	\$0.00/hr \$0.00/yr	\$0.105/hr	\$0.105/hr
c4.2xlarge	\$0.00/hr \$0.00/yr	\$0.419/hr	\$0.419/hr
c4.xlarge	\$0.00/hr \$0.00/yr	\$0.838/hr	\$0.838/hr
c4.8xlarge	\$0.00/hr \$0.00/yr	\$1.675/hr	\$1.675/hr
r3.xlarge	\$0.00/hr \$0.00/yr	\$0.333/hr	\$0.333/hr
r3.2xlarge	\$0.00/hr \$0.00/yr	\$0.665/hr	\$0.665/hr
r3.4xlarge	\$0.00/hr \$0.00/yr	\$1.33/hr	\$1.33/hr
r3.8xlarge	\$0.00/hr \$0.00/yr	\$2.66/hr	\$2.66/hr

EBS General Purpose (SSD) volumes
\$0.10 per GB-month of provisioned storage

Assumes On-Demand EC2 pricing; prices for Reserved and Spot Instances will be lower. See [pricing details](#).

Data transfer fees not included.

[Learn about instance types](#)

Click on the button marked **Launch with EC2 console** to start a single-instance of Alces Flight Compute in the region of your choice. Alternatively, you can note the AMI-ID and use this to manually configure your AMI in the EC2 console. Launch your instance with the settings you require, choosing the instance type, disk size and security group settings suitable for your environment. Note - your security group needs to allow port 22 (SSH) to be accessed from your client system to allow you to login and configure the instance for use.

Note: We recommend that instances launching Alces Flight Compute are provided with a root EBS volume of at least 20GB in size to allow for storage of application data.

Accessing and configuring your instance

Once launched, your Alces Flight Compute instance will allow SSH login to the public IP address provided with your instance. Use your SSH client to login to the instance, using the default username `alces` and the SSH keypair you provided at launch; e.g. for a instance with IP address `52.51.18.191`, use the command:

```
ssh alces@52.51.18.191
```

After logging in to the environment, users will be notified that the instance is not yet configured:



```

52.51.18.191 - PuTTY
./oooo+`
~/ooooooo.
/ooooooo/
ooooooo- ./o/      Alces Clusterware (r2016.2r3-rc2)
+ooooooo/`+ooo    Based on CentOS Linux 7.2.1511 (Core)
-oooooooooooooo
:ooooooooooooo. `:+:~
-+oooooooooooo+:ooo`
`:ooooooooooooooo.
`:+ooooooooooooo+`
`-+oooooooooooo+-
.:+oooooooooooo+~
`-/oooooooooo/..-....-:/+ooo//oooo+/+oooooo/
./ooooooooooo+oooooooooooooooooooooooooooooo/`
.+oooooooooooo+oooooooooooooooooooooooooooo+:.
.:/+oooooooo-`..--:~::~:-.-`
`-:/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

=====
PLEASE NOTE
=====

Configuration of this node has not yet been completed and it is not yet
operational.

Please proceed with configuration by running the "alces configure" command.

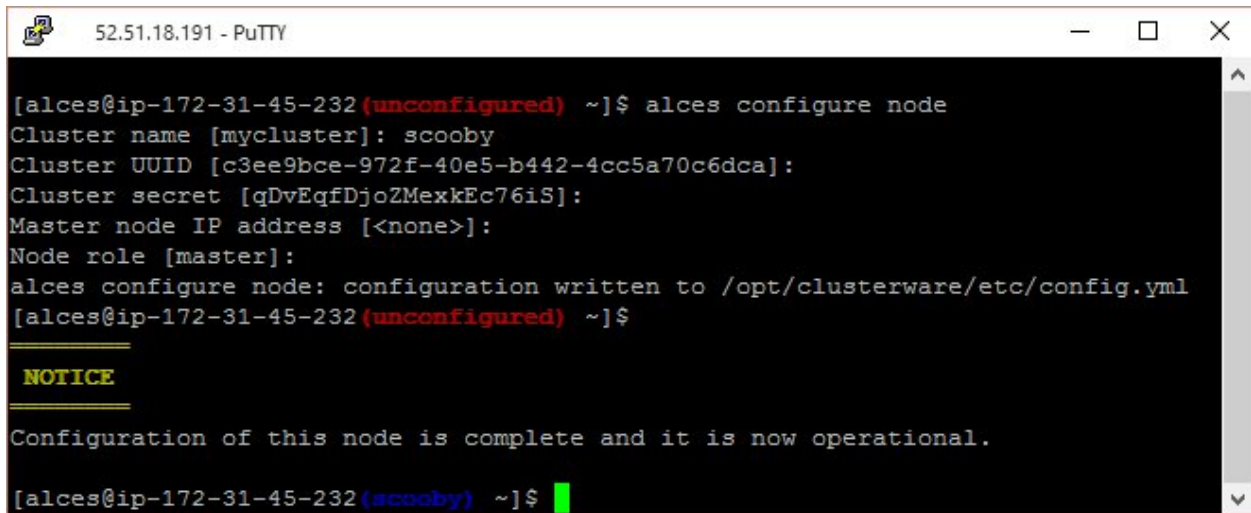
Generating SSH keypair: OK
Authorizing key: OK
[alces@ip-172-31-45-232 (unconfigured) ~]$

```

Use the `alces configure node` command to setup the instance, answering the following questions:

- **Cluster name:** The name to call your compute cluster (up to 128 characters long).
- **Cluster UUID:** The unique ID of your cluster. If the instance you are configuring will be the first (or only) node in your cluster, accept the default provided.
- **Cluster secret:** The secret passphrase for your cluster. If the instance you are configuring will be the first (or only) node in your cluster, accept the default provided.
- **Master node IP address:** Enter the IP address of your cluster login node, if you already have one.
- **Node role:** Accept the default (`master`) if this instance will be your cluster login node or enter `slave` to configure a compute node instance.

After a few seconds, the instance will be configured in the target role requested - users are notified at the command prompt once configuration is complete.

A screenshot of a PuTTY terminal window titled "52.51.18.191 - PuTTY". The terminal shows the execution of the `alces configure node` command. The user provides the cluster name "scooby", a UUID, a secret, and the master node IP address. The terminal confirms the configuration is written to `/opt/clusterware/etc/config.yml`. A yellow "NOTICE" banner appears, stating "Configuration of this node is complete and it is now operational." The prompt then changes from `(unconfigured)` to `(scooby)`.

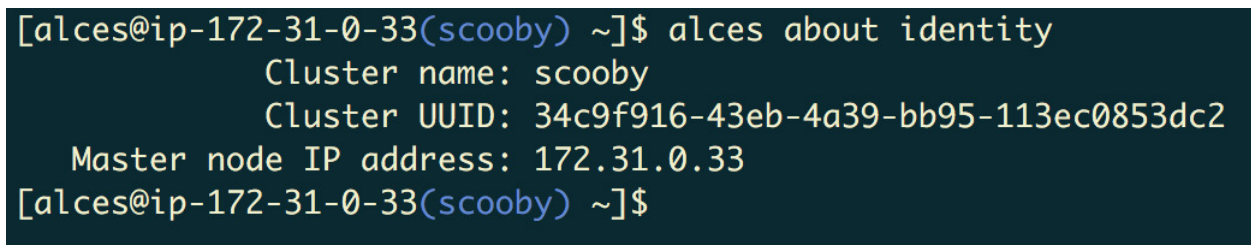
```
[alces@ip-172-31-45-232 (unconfigured) ~]$ alces configure node
Cluster name [mycluster]: scooby
Cluster UUID [c3ee9bce-972f-40e5-b442-4cc5a70c6dca]:
Cluster secret [qDvEqfDjoZMexkEc76iS]:
Master node IP address [<none>]:
Node role [master]:
alces configure node: configuration written to /opt/clusterware/etc/config.yml
[alces@ip-172-31-45-232 (unconfigured) ~]$

NOTICE

Configuration of this node is complete and it is now operational.

[alces@ip-172-31-45-232 (scooby) ~]$
```

After configuration is complete, you can use the `alces about node` and `alces about identity` commands to recall the configuration information about your instance to allow further compute nodes to be added to an existing cluster:

A screenshot of a terminal window showing the output of the `alces about identity` command. The output displays the cluster name, UUID, and master node IP address.

```
[alces@ip-172-31-0-33(scooby) ~]$ alces about identity
Cluster name: scooby
Cluster UUID: 34c9f916-43eb-4a39-bb95-113ec0853dc2
Master node IP address: 172.31.0.33
[alces@ip-172-31-0-33(scooby) ~]$
```

You can find the cluster secret token in `/opt/clusterware/etc/config.yml` - you will need the secret token to configure any additional hosts.

Adding more nodes to your cluster

Once you have configured a master node using the method above (i.e. a login node) customised compute clusters can be constructed by starting further individual instances or groups of instances of different types; these can use different charging methods (on-demand/reserved/spot) or be placed in different availability zones.

To proceed with adding compute nodes to your cluster, launch further instances through the EC2 console and configure them as slave nodes (i.e. compute nodes) by using the `alces configure node` command and specifying the `slave node` role.

Note: When configuring a slave node, you must supply the same **Cluster UUID** and **Cluster secret** values specified for other nodes in the cluster or the instance will be unable to communicate with the rest of the cluster.

Note that you may use your own autoscaling group configuration or simply manually launch individual instances.

Launching Alces Flight on AWS using a CloudFormation template

Using AWS Marketplace is a convenient method to launch your Alces Flight cluster, using a pre-defined template that collect common configuration options for an HPC compute cluster. For advanced users, it is also possible to launch individual instances from the Alces Flight AMI - for more details, see - [Launching a single Alces Flight instance on AWS](#).

Users can also create their own CloudFormation templates to launch different cluster configurations. While a base knowledge of CloudFormation is required, this method is often preferable to configuring individual instances as it allows clusters to be repeatably launched once a customized template has been created.

Alces provides a number of example templates that are intended to assist users in generating their own templates. The templates below use the Flight AMI from AWS Marketplace to build your cluster - users are encouraged to review these templates with an aim to launching their own, customised environments.

Example AWS CloudFormation templates

8-node, 16-node and 32-node fixed-size clusters

- <https://s3.amazonaws.com/alces-flight-templates/2016.2r3/8-node.json>
- <https://s3.amazonaws.com/alces-flight-templates/2016.2r3/16-node.json>
- <https://s3.amazonaws.com/alces-flight-templates/2016.2r3/32-node.json>

This template is designed to create an 8-node cluster, and provides a choice of compute and login node instance types. The template creates:

- A VPC, subnet and gateway for the cluster
- An on-demand login node with EBS (Magnetic) storage
- 8, 16 or 32 compute nodes of on-demand or spot type in an autoscaling group

Variable sized cluster

- <https://s3.amazonaws.com/alces-flight-templates/2016.2r3/x-node.json>

This template is designed to create a cluster of between 1 and 32 nodes, and provides a choice of compute and login node instance types. The template creates:

- A VPC, subnet and gateway for the cluster

- An on-demand login node with EBS (Magnetic) storage
- A choice of compute nodes of on-demand or spot type in an autoscaling group

Demonstration cluster

- <https://s3.amazonaws.com/alces-flight-templates/2016.2r3/demo.json>

This template is designed to create a cluster of between 1 and 32 nodes, with higher performance SSD-backed EBS storage (sg2) and all software repositories enabled for demonstration purposes. The template creates:

- A VPC, subnet and gateway for the cluster
- An on-demand login node with SSD-backed EBS (sg2) storage
- A choice of compute nodes of on-demand type
- Both the main and volatile software repositories enabled

Disabling resource limits for your job-scheduler

By default, an Alces Flight Compute cluster enables limits on the resources your jobs can use on the compute nodes of your cluster. These limits are designed to help protect the compute nodes from being overloaded, and help to promote efficient usage of your compute cluster. It is important to train HPC cluster users in requesting the right resources for their jobs, as this is a fundamental principle when working on shared HPC facilities, where users do not have the benefit of exclusive use.

Experienced HPC cluster users working on a personal Alces Flight Compute cluster may prefer to disable the job-scheduler resource limits and manage compute node resources themselves. These instructions provide a guide to disabling the default limits for your compute cluster job scheduler.

Note: Once scheduler resource limits are disabled, users must manually ensure that the compute node instances selected have enough resources to run the jobs submitted to the job-scheduler queue.

Open-grid scheduler

By default, your Alces Flight Compute cluster sets limits for the following properties of your compute jobs:

- Number of CPU cores
- Amount of memory per requested slot (measured per CPU core, or complete node)
- Run time (measured in seconds of node occupancy)

Requesting more CPU cores for your job

The default number of CPU cores for your job is one. This setting is used in the absence of a Parallel Environment (PE) request and cannot be sensibly over-ridden. Users can however the job-scheduler to request a default parallel-environment by adding a job-scheduler instruction into their local `$HOME/.sge_request` file. For example; if a user wanted to ensure that all jobs requested the **SMP** parallel environment with 2 CPU-cores, they could add the following into their `$HOME/.sge_request` file:


```
-pe smp 2
```

Note: Job-scheduler instructions provided as parameters to the `qsub` command, or included in job-scripts are prioritised over the contents of your `$HOME/.sge_request` file.

Use the command `man sge_request` for more information on configuring default settings for job submitted on your compute cluster.

Disabling memory resource limits

Users can follow the process below to disable memory limits for jobs running via the cluster job-scheduler.

1. Use the `sudo -s` command on the cluster login node to become the root user.
2. Run the command `EDITOR=nano qconf -mc`
3. Using the **nano** editor, change the settings for the `h_vmem` entry so that:
 - requestable is set to **NO**
 - consumable is set to **NO**
 - default is set to **0** (zero)
4. Press **CTRL+X** to exit the nano editor

Once memory limits are disabled, cluster users can submit jobs without a `-l h_vmem` setting, allowing jobs to use as much memory as is available on their cluster compute nodes.

Note: A job submitted via the job-scheduler that uses more memory than your compute node can provide may be automatically killed by Linux, or could in extreme cases cause your compute node to stop responding. Users must manually manage memory resources if memory resource limits are disabled in the cluster job scheduler.

Disabling the default runtime limit

Submitted jobs automatically have a default runtime applied, unless overridden with the `-l h_rt=` parameter to `qsh` or `qsub` on the command-line, or included as a job-scheduler instruction in your job-script.

Users can optionally disable the default runtime limit for their cluster by following these steps:

1. Edit the system-wide grid-scheduler defaults file; for example, to use the nano editor, use the command:

```
nano $GRIDSCHEDULERDIR/etc/common/sge_request
```
2. Remove the `h_rt=24:00:00` section at the end of the file
3. Save the file by pressing **CTRL+X**

Once the default runtime limit is disabled, cluster users can submit jobs without a `-l h_rt=` setting, allowing jobs to run forever until they complete or are terminated by the user.

Note: A job submitted via the job-scheduler without a run-time limit will run forever until it naturally completes or is terminated by the user. Users should review their jobs to ensure that suitable infrastructure has been selected for long-running jobs, and may wish to setup billing alerts via their platform provider to ensure that run-away jobs are identified and stopped promptly.

Alces Gridware Software Applications

This page documents the software which is currently available via the Alces Gridware project. Software applications are listed in the Alces Gridware repository with the structure `repository/type/name/version`, which corresponds to:

- **repository** - packages are listed in the **main** repository if available for auto-scaling clusters, 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

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.

The following list is updated periodically as new software is added to the repositories. For clarity, multiple versions of the same application are not shown, unless there are significant functional differences between versions. Software categories are provided as a guide only - many packages are multi-discipline and have a range of appropriate uses.

Tools and Utilities

gnuplot	A portable command-line driven graphing utility
grace	Grace is a WYSIWYG 2D plotting tool for the X Window System and M*tif
idr	Framework to measure the reproducibility of findings
mawk	mawk is an interpreter for the AWK Programming Language.
parallel	A shell tool for executing jobs in parallel using one or more_
↪computers	

Benchmarks

HPL	A Portable Implementation of the High-Performance Linpack Benchmark_
↪ for	Distributed-Memory Computers
memtester	A userspace utility for testing the memory subsystem for faults.
hpcc	The HPC Challenge benchmark
imb	IMB is a networking benchmark tool.
iozone	IOzone is a filesystem benchmark tool.
gpuburn	GPU-Burn is a stress test for multi-GPGPU-setups.
Folding@home	A distributed computing project studying protein folding and _
↪misfolding	

Biochemistry

amber	Set of programs for biomolecular simulation and analysis
beast	Cross-platform program for Bayesian MCMC analysis of molecular_
↪sequences	
placement	An algorithm for prediction of explicit solvent atom distribution

BioInformatics

MS Prot Tools	Some hopefully useful tools for mass spectrometry applied to_
↳proteomics	
454 Sequencing	454 Sequencing System Off-Instrument Software Applications suite
ABRA	Assembly Based ReAligner
'AbySS	A de novo, parallel, paired-end sequence assembler that is designed_
↳for short reads	
ANGES	Reconstructs ancestral genome maps from homologous markers in extant_
↳related genomes	
ANNOVAR	Functional annotation of genetic variants from high-throughput_
↳sequencing data	
ARAGORN	Detect tRNA genes and tmRNA genes in nucleotide sequences
ArtificialFqG	Outputs artificial FASTQ files derived from a reference genome
Atlas-SNP2	Next-generation sequencing suite of variant analysis tools
AUGUSTUS	Predicts genes in eukaryotic genomic sequences
autoadapt	Automatically detect and remove adaptors and primers present in a_
↳FASTQ file	
bam-readcount	Generate metrics at single nucleotide positions
BAMStats	Tool to calculate and display various metrics derived from SAM/BAM_
↳files	
BamTools	Provides a programmer's API and an end-user's toolkit for handling_
↳BAM files	
bamUtil	Several programs that perform operations on SAM/BAM files
BamView	Interactive display of read alignments in BAM data files
BLAST	Compares nucleotide or protein sequences to sequence databases and_
↳calculates the statistical significance of matches	
Batalign	An incremental method for accurate gapped alignment
BCftools	Utilities for variant calling and manipulating VCF and BCF files
Bcl2FastQ	A tool to handle bcl conversion and demultiplexing
Bcl2FastQ	A tool to handle bcl conversion and demultiplexing
BCL Converter	Convert *.bcl files into *_qseq.txt files
BEAGLE	A general purpose library for evaluating the likelihood of sequence_
↳evolution on trees	
BEAGLE	Analysis of large-scale genetic data sets with hundreds of thousands_
↳of markers genotyped on thousands of samples	
BEAGLE	A general purpose library for evaluating the likelihood of sequence_
↳evolution on trees	
BEDTools	A flexible suite of utilities for comparing genomic features
biom-format	The Biological Observation Matrix (BIOM) format
BIONJ	An improved version of the NJ algorithm based on a simple model of_
↳sequence data	
BioPerl	A community effort to produce Perl code which is useful in biology
Biopython	Set of freely available tools for biological computation written in_
↳Python	
bio-rainbow	Package for RAD-seq related clustering and de novo assembly.
Bismark	A bisulfite read mapper and methylation caller
BLAST (Legacy)	Compares nucleotide or protein sequences to sequence databases and_
↳calculates the statistical significance of matches	
Bowtie 2	Fast and sensitive read alignment
Bowtie	Ultrafast memory-efficient short read aligner
BreakPointer	Pinpoint rearrangement breakpoints using paired end reads
CAP3	Sequence Assembly Program
car	Reconstructing contiguous regions of an ancestral genome
cdbfasta	CDB (Constant DataBase) indexing and retrieval tools for FASTA files
CD-HIT	A program for clustering DNA/protein sequence database at high_
↳identity with tolerance.	
cdhit	A program for clustering and comparing protein or nucleotide_
↳sequences.	

CEGMA	Building sets of gene annotations in eukaryotic genomes
CGAT	The Computational Genomics Analysis Toolkit
CHANCE	Assess the quality of ChIP-seq experiments
CHIAMO	Call genotypes from the Affymetrix 500K Mapping chip
Chimerascan	Detection of chimeric transcripts in high-throughput sequencing data
ClonalFrame	Inference of bacterial microevolution using multilocus sequence data
CLUMPP	Deals with label switching and multimodality problems in population- ↳genetic cluster analyses
Clustal Omega	Multiple alignment of nucleic acid and protein sequences
ClustalW	Multiple alignment of nucleic acid and protein sequences
cnD	Copy number variant caller for inbred strains
CNVnator	CNV discovery and genotyping from depth of read mapping
CoNIFER	Copy Number Inference From Exome Reads
CASAVA	Processes sequencing reads provided by RTA or OLB
CONTIGuator	A bacterial genomes finishing tool for structural insights on draft_ ↳genomes
Control-FREEC	Detect copy-number changes and allelic imbalances using deep- ↳sequencing data
CNATR	Tool for copy number variation (CNV) detection for targeted_ ↳resequencing data
Corset	Software for clustering de novo assembled transcripts and counting_ ↳overlapping reads
Cortex	"Software for genome assembly and variation analysis
CRAMTools	Set of Java tools and APIs for efficient compression of sequence read_ ↳data
CREST	Algorithm for detecting genomic structural variations at base-pair_ ↳resolution
CRISP	Multi-sample variant caller for high-throughput pooled sequence data
Curtain	Assembler of next generation sequence, developed by Matthias Haimel_ ↳in the Ensembl Genomes team at the EBI
cutadapt	A tool that removes adapter sequences from DNA sequencing reads
DARWIN	Data Analysis and Retrieval With Indexed Nucleotide/peptide sequences
DDiMAP	Analyses mapped NGS read data to discover rare variants
dDocent	An interactive bash wrapper to QC, assemble, map, and call SNPs from_ ↳double digest RAD data
Delly	Structural variant discovery by integrated paired-end and split-read_ ↳analysis
distruct	Graphically display results produced by the genetic clustering_ ↳program structure
DREEP	Detecting low-level mutations by utilizing the RE-sequencing Error_ ↳Profile of the data
EAD	Error aware demultiplexer is a probabilistic demultiplexer for_ ↳Illumina BCL files.
EIGENSOFT	Combines functionality from population genetics methods and_ ↳EIGENSTRAT stratification method
EIGENSOFT	Combines functionality from population genetics methods and_ ↳EIGENSTRAT stratification method
Ensembl API	Abstraction layer for accessing Ensembl genomic databases
Ensembl Variant Effect Predictor	Predict the functional consequences of known_ ↳and unknown variants
e-PCR	Identifies sequence tagged sites (STSs) within DNA sequences
Exonerate	Generic tool for pairwise sequence comparison
eXpress	Streaming tool for quantifying the abundances of a set of target_ ↳sequences from sampled subsequences
FamSeq	A computational tool for calculating probability of variants in_ ↳family-based sequencing data
FASTA	Search protein or DNA sequence databases comparing a protein sequence_ ↳to a DNA sequence database

FastQC	A quality control tool for high throughput sequence data
fastq-tools	Small utilities for working with fastq sequence files
FastTree	Inference of approximately-maximum-likelihood phylogenetic trees from ↪alignments of nucleotide or protein sequences.
FastUniq	an ultrafast de novo duplicates removal tool for paired short DNA ↪sequences
Flexbar	Flexible barcode and adapter removal for sequencing platforms
FreeBayes	Bayesian genetic variant detector designed to find small polymorphisms
FREGENE	Simulates sequence-like data over large genomic regions in large ↪diploid populations
FusionFinder	Find fusion transcript candidates in RNA-Seq data
FusionMap	Align reads spanning fusion junctions directly to the genome
Galaxy	Open, web-based platform for data intensive biomedical research
GBrowse	The Generic Genome Browser
geneid	Predicts genes in anonymous genomic sequences designed with a ↪hierarchical structure
GeneTorrent	Transfer genomic data reliably across a network
GATK	Software package developed at the Broad Institute to analyse next- ↪generation resequencing data
GAT + queue	Broad Institute package for analysing next-generation resequencing ↪data; including command-line scripting framework for defining multi-stage genomic ↪analysis pipelines
GenomeMapper	Short read mapping tool designed for accurate read alignments
GENSCAN	Analyze genomic DNA sequences from a variety of organisms
GERP++	Identifies constrained elements in multiple alignments by quantifying ↪substitution deficits
GIMSAN	GIbbsMarkov with Significance Analysis
Glimmer	System for finding genes in microbial DNA, especially the genomes of ↪bacteria, archaea, and viruses
GMAP/GSNAP	Genomic mapping and alignment and short-read nucleotide alignment ↪programs
GREAT	Genomic Regions Enrichment of Annotations Tool
Grinder	A versatile omics shotgun and amplicon sequencing read simulator
GTOOL	Transforms sets of genotype data for use with the programs SNPTEST ↪and IMPUTE
HAL	Hierarchical Alignment Format API and analysis and conversion tools
hapflk	Haplotype-based test for differentiation in multiple populations
HLA*IMP BE	Impute HLA type information based on SNP genotypes back-end
HLA*IMP FE	Impute HLA type information based on SNP genotypes front-end
HMMcopy	Make copy number estimations for whole genome data
HTPcall	Improved base-calling for homopolymer-sensitive next-gen data
HTSeq	Process data from high-throughput sequencing assays
HTSlib	C library for high-throughput sequencing data formats
abacas	Rapidly contiguate (align, order, orientate), visualize and design ↪primers
Bio-bwa	Aligns relatively short nucleotide sequences against a long reference ↪sequence such as the human genome
bowtie2	Fast and sensitive read alignment
bowtie	Ultrafast memory-efficient short read aligner
bedtools	A flexible suite of utilities for comparing genomic features
phast	Software package for comparative and evolutionary genomics
cufflinks	Assembles transcripts, estimates their abundances, and tests for ↪differential expression and regulation in RNA-Seq samples
emboss	Software analysis suite developed for the molecular biology community
genetics	Reports position-specific measures of conservation
genome	An alignment tool like BLAST
varscan	Mutation caller for targeted, exome, and whole-genome resequencing ↪data

breakdancer	Provides genome-wide detection of structural variants from next-generation paired-end sequencing reads
Genome-music	A comprehensive analysis suite for mutations in cancer genomes
radmarkers	Guppy RAD tools
fastx	A collection of command line tools for Short-Reads FASTA/FASTQ files
	↳ preprocessing
fastx2	Assaf Gordon text utilities
hmmer	Biosequence analysis using profile hidden Markov models
htsfilter	Standard Filter for identification of polyclonal and independant errors for SOLiD short read sequences
macs	Novel algorithm for identifying transcript factor binding sites
sambamba	Tools for working with SAM/BAM data
mag	Builds qassembly by mapping short reads to reference sequences
picard	Java-based command-line utilities and API for manipulating SAM files
ribopicker	Identify and remove rRNA sequences from metagenomic and metatranscriptomic datasets
samtools	Provides various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format
plinkseq	Library for working with human genetic variation data
bamview	Variant detector and alignment viewer for next-generation sequencing data in the SAM/BAM format
recon	Package for finding repeat families from biological sequences
Picard-broad	Command line tools for manipulating high-throughput sequencing (HTS) data and formats
mabkit	Tools for common BAM file manipulations
speedseq	A flexible framework for rapid genome analysis and interpretation
fgwas	Functional genomics and genome-wide association studies
lighter	Fast and memory-efficient sequencing error corrector
bamtools3	Provides a programmer's API and an end-user's toolkit for handling BAM files
diffreps	Differential analysis for ChIP-seq with biological replicates
Macs-taoliu	Novel algorithm for identifying transcript factor binding sites
sift	Predicts whether an amino acid substitution affects protein function
impute	A genotype imputation and phasing program based on ideas from Howie et al. (2009)
snpmatic	Fast, stringent short-read mapping software
soap	A short read de novo assembly tool
amos	A collection of tools and class interfaces for the assembly of DNA reads
bfast	Facilitates the fast and accurate mapping of short reads to reference sequences
kggseq	A biological knowledge-based mining platform for genomic and genetic studies using sequence data
passion	A pattern growth algorithm based pipeline for splice site detection in paired-end RNA-Seq data
Cnv-seq	A method for detecting DNA copy number variation (CNV) using highthroughput sequencing
tophat	A spliced read mapper for RNA-Seq
Tophat-fusion	Enhanced version of TopHat with the ability to align reads across fusion points
trinitymaseq	A novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data
vcftools	Package designed for working with VCF files, such as those generated by the 1000 Genomes Project
fastq_screen	A screening application for high throughput sequence data
gatk	Software package developed at the Broad Institute to analyse next-generation resequencing data

igv	A high-performance visualization tool for interactive exploration of ↵ ↵large, integrated genomic datasets
scripture	Java-based command-line tool for transcriptome reconstruction
bertone	Subdivision of ChIP-seq/ChIP-chip regions into discrete signal peaks
oases	De novo transcriptome assembler for very short reads
velvet	Sequence assembler for very short reads
hgsc	SNP discovery tool developed for next generation sequencing platforms
illuminautils	File utilities for Illumina sequencers
htslib	Provides various utilities for manipulating alignments in the SAM ↵ ↵format, including sorting, merging, indexing and generating alignments in a per- ↵position format
bam2fastq	Extract raw sequences (with qualities)
mothur	Provides microbial ecologists with the functionality of dotur, sons, ↵ ↵treeclimber, s-libshuff, unifrac and more.
mutationtaster	Next-generation sequencing pipeline from Mutation Taster (http
ngsqctoolkit	A toolkit for the quality control (QC) of next generation sequencing ↵ ↵(NGS) data
w.cgi	Software for performing Bayesian inference Using Gibbs Sampling
repeatmasker	Screens DNA sequences for interspersed repeats and low complexity DNA ↵ ↵sequences
artemis	Genome browser and annotation tool
dindel	Calls small indels from next-generation sequence data by realigning ↵ ↵reads to candidate haplotypes
reapr	Evaluates the accuracy of a genome assembly using mapped paired end ↵ ↵reads
smalt	Efficiently aligns DNA sequencing reads with genomic reference ↵ ↵sequences
shapeit	Segmented HAPlotype Estimation and Imputation Tool - Fast and ↵ ↵accurate haplotype inference
stampy	Maps short reads from Illumina sequencing machines on to a reference ↵ ↵genome
iassembler	Assemble ESTs generated using Sanger and/or Roche-454 pyrosequencing ↵ ↵technologies into contigs
Infernal	Search DNA sequence databases for RNA structure and sequence ↵ ↵similarities
InterProScan	Allows sequences to be scanned against InterPro's signatures
JAGS	Analysis of Bayesian hierarchical models using Markov Chain Monte ↵ ↵Carlo simulation
Kent src utils	Jim Kent and the UCSC Genome Bioinformatics Group program suite
khmer	k-mer counting, filtering and graph traversal
LASTZ	Program for aligning DNA sequences, a pairwise aligner
LifeScope	LifeScope Genomic Analysis Solutions Tools
LOCAS	Low-coverage short-read assembler
LoFreq	Fast and sensitive variant-caller for inferring SNVs from high- ↵throughput sequencing data
LUMPY	A probabilistic framework for structural variant discovery
MAFFT	Multiple alignment program for amino acid or nucleotide sequences
mafJoin	Tool for combining pairs of maf files that share a common sequence
MAKER	Portable and easily configurable genome annotation pipeline
M.A.Q Viewer	Graphical read alignment viewer
MaSuRCA	Whole genome assembly
Mauve	Mauve Genome Alignment Software
MeDUSA	Methylated DNA Utility for Sequence Analysis - Computational pipeline ↵ ↵to perform a full analysis of MeDIP-seq data
MEGA	Molecular Evolutionary Genetics Analysis - Software suite for ↵ ↵analyzing DNA and protein sequence data from species and populations
MEME Suite	Motif-based sequence analysis tools
MERLIN	Uses sparse trees to represent gene flow in pedigrees

Microbiome	Microbiome Utilities
MIRA	Whole genome shotgun and EST sequence assembler
MISO	Probabilistic analysis and design of RNA-Seq experiments for_
↪identifying isoform regulation	
MitoSeek	Extraction of mitochondrial genome information from exome sequencing_
↪data	
MODELLER	Program for Comparative Protein Structure Modelling by Satisfaction_
↪of Spatial Results	
mpiBLAST	Open-Source Parallel BLAST
MrBayes	Bayesian Inference of Phylogeny
Multiz/TBA	Threaded-Blockset Aligner, a local multiple sequence alignment tool;_
↪MULTIZ, aligns highly rearranged or incompletely sequenced genomes	
MUMmer	System for rapidly aligning entire genomes
MUSCLE	Multiple sequence alignment
MuTect	Reliable and accurate identification of somatic point mutations
NGS-SNP	Collection of command-line scripts for providing rich annotations for_
↪SNPs	
Novoalign	Aligner for short nucleotide space reads
NucleoATAC	Package for calling nucleosomes using ATAC-Seq data
Oases	De novo transcriptome assembler for very short reads
454-OISA	454 Sequencing System Off-Instrument Software Applications suite
OL Basecaller	Performs base calling and bcl to qseq conversion for the HiSeq,_
↪HiScan-SQ, or Genome Analyzer	
ONCOCNV	Detection of copy number changes in Deep Sequencing data
OncoSNP-SEQ	Characterise copy number alterations and loss-of-heterozygosity events
Oncotator	Annotate human genomic point mutations and indels with data relevant_
↪to cancer researchers	
OpenMS	LC/MS data management and analyses
PAGIT	Post Assembly Genome Improvement Toolkit
PAML	Phylogenetic Analysis by Maximum Likelihood
Panseq	Determine the core and accessory regions among a collection of_
↪genomic sequences	
PeakRanger	Multi-purpose software suite for analyzing next-generation_
↪sequencing (NGS) data	
PEAR	PEAR is an ultrafast, memory-efficient and highly accurate pair-end_
↪read merger	
PeSV-Fisher	Pipeline for the detection of five general types of structural_
↪variants	
phantompeakqual	Compute quick, highly informative enrichment and quality measures for_
↪ChIP-seq/DNase-seq/FAIRE-seq/MNase-seq data	
phrap	phrap is a program for assembling shotgun DNA sequence data
PHYLP	A free package of programs for inferring phylogenies
Pindel	Detection of breakpoints of structural variants at single-based_
↪resolution from next-gen sequence data	
plink	Whole genome association analysis toolkit
Polymutt	Calls single nucleotide variants and detects de novo point mutation_
↪events in families for next-generation sequencing data	
PolyPhen-2	Predicts possible impact of amino acid substitutions on the structure_
↪and function of human proteins	
popoolation2	Allows comparison of allele frequencies between two ore more_
↪populations	
popoolation	Estimate natural variation and positive selection
Preseq	Predict and estimate the complexity of a genomic sequencing library
Primer3	PCR primer design tool
PRINSEQ Lite	Filter, reformat, or trim genomic and metagenomic sequence data
PROCHECK	Stereochemical protein structure quality analysis
ProgCactus	A whole-genome alignment package
PSIPRED	Accurate protein secondary structure prediction

PyCogent	A toolkit for making sense from sequence
PyNAST	Python Nearest Alignment Space Termination tool
pyprophet	Analyse MRM data
PyroBayes	A novel base caller for pyrosequences from the 454 Life Sciences_
↪sequencing machines	
Q	Whole genome association analysis toolkit
Qiime	Quantitative Insights Into Microbial Ecology
RAxML	Randomized Axelerated Maximum Likelihood Sequential and_
↪parallel inference of large phylogenies with maximum likelihood	
Rcount	Simple and flexible RNA-Seq read counting
RDP Classifier	Naive Bayesian classifier that can rapidly and accurately provides_
↪taxonomic assignments from domain to genus.	
RepeatNet	An ab initio centromeric sequence detection algorithm
rMATS	Detect differential alternative splicing events from RNA-Seq data
RMBlast	NCBI Blast modified for use with RepeatMasker/RepeatModeler
RSEM	Estimate gene and isoform expression levels from RNA-Seq data
RSeQC	An RNA-seq Quality Control Package
RStudio Desktop	A free and open source integrated development environment for R
samblaster	Mark duplicates and extract discordant and split reads from sam files
Satsuma	High-sensitivity alignments through cross-correlation
screed	Short read sequence utils in Python.
Scythe	A very simple adapter trimmer
Scythe	A very simple adapter trimmer
SeqAn	Open source C++ library of efficient algorithms and data structures_
↪for the analysis of sequences	
SeqClean	The Gene Indices Sequence Cleaning and Validation script (SeqClean)
SeqEM	Adaptive genotype-calling approach for next-generation sequencing_
↪studies	
SeqGene	Software for mining next-gen sequencing datasets
Seqtk	Toolkit for processing sequences in FASTA/Q formats
SRAT	Programmatically access data housed within SRA and convert it from_
↪the SRA format	
SSAHA	Sequence Search and Alignment by Hashing Algorithm - A pairwise_
↪sequence alignment program for efficient mapping of sequencing reads	
SVA	Sequence Variant Analyzer - Annotate, visualize and analyze the_
↪genetic variants indentified through next-generation sequencing studies	
SOAP	Short Oligonucleotide Analysis Package - An updated version of SOAP_
↪software for short oligonucleotide alignment	
SHRiMP	Software package for aligning genomic reads against a target genome
SICER	Identify enriched domains from histone modification ChIP-Seq data
Sickle	A windowed adaptive trimming tool for FASTQ files using quality
Sickle	A windowed adaptive trimming tool for FASTQ files using quality
Sierra Perl	Perl client to access Sierra, the Stanford HIV Web Service
SiPhy	Rigorous statistical tests to detect bases under selection from a_
↪multiple alignment data	
SNAP	General purpose gene finding for both eukaryotic and prokaryotic_
↪genomes	
snpEff	Fast variant effect predictor (SNP, MNP and InDels) for genomic data
SNPTEST v2	Analysis of single SNP association in genome-wide studies
SNVMix	Detect single nucleotide variants from next generation sequencing data
SOAPdenovoTrans	A de novo transcriptome assembler designed specifically for RNA-Seq
SOAPfusion	Fusion discovery with paired-end RNA-Seq reads
SOAP-ICLU	Identify genome-wide large variants, such as CNVs and LOH etc.
'SOAPIndel'	Call indels from next-generation paired-end sequencing data
SOAPSnp	Calls consensus genotype by carefully considering data quality,_
↪alignment and recurring experimental errors	
'SOAPsplice'	Genome-wide ab initio detection of splice junction sites from RNA-Seq
STIR	Software for Tomographic Image Reconstruction - Multi-platform object-
↪oriented framework for data manipulations in tomographic imaging	

SortMeRNA	SortMeRNA is a software designed to rapidly filter ribosomal RNA fragments from metatranscriptomic data produced by next-generation sequencers.
Stacks	Software pipeline for building loci from short-read sequences
Staden Package	A fully developed set of DNA sequence assembly (Gap4 and Gap5), editing and analysis tools (Spin)
STAR	Aligns RNA-seq reads to a reference genome using uncompressed suffix arrays
Strelka	Somatic variant calling workflow for matched tumor-normal samples
StructHarvester	Extracting data from STRUCTURE results files
Structure	Use multi-locus genotype data to investigate population structure
SuperHirn	Tool to quantitatively analyze multi-dimensional LC-MS data
SVMerge	Enhanced structural variant and breakpoint detection
Tabix++	C++ wrapper to Tabix indexer for TAB-delimited genome position files
Tabix	Generic indexer for TAB-delimited genome position files
Tablet	Lightweight, high-performance graphical viewer for next generation sequence assemblies and alignments
Tandem Repeats	Locate and display tandem repeats in DNA sequences
tax2tree	Assists in decorating an existing taxonomy onto a phylogenetic tree with overlapping tip names
T-Coffee	Align sequences or combine the output of other alignment methods into one unique alignment
TMAP	Torrent mapping alignment program
TopHat	A spliced read mapper for RNA-Seq
Trans-ABYSS	Analyze ABYSS multi-k-assembled shotgun transcriptome data
treeviewx	Phylogeny tree viewer
Trim Galore!	Wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files
Trimmomatic	A flexible read trimming tool for Illumina NGS data
Trinity	A novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data
trRNAscan-SE	Improved detection of transfer RNA genes in genomic sequence
UNPHASED	Software for genetic association analysis
USeq	Collection of software tools for analysis of sequencing data from the Solexa, SOLiD, and 454 platforms
VariationHunter	A tool for discovery of structural variation in one or more individuals simultaneously using high throughput technologies
vcflib utils	Command-line utilities for executing complex manipulations on VCF files
Velvet	Sequence assembler for very short reads
VerifyBamID	Verify whether reads match previously known genotypes for an individual
Vienna RNA	RNA Secondary Structure Prediction and Comparison
Vmatch	Software tool for efficiently solving large scale sequence matching tasks
Wise2	Program for aligning proteins or protein HMMs to DNA
WU BLAST	Washington University-produced alternative to NCBI BLAST
wwwblast	A suite of standalone BLAST programs produced by NCBI for use on the web
Bioconductor	Tools for the analysis and comprehension of high-throughput genomic data

Bio-physics

vmd	Molecular visualization program for displaying, animating, and analyzing large biomolecular systems
molscript	MolScript is a program for displaying molecular 3D structures

NAMD	A parallel molecular dynamics code designed for high-performance_
↳simulation of	large biomolecular systems
PyMOL	PyMOL is a Python-enhanced molecular graphics tool
RasMOL	RasMol is a program for molecular graphics visualisation

Chemistry

ASE	Atomistic Simulation Environment - Python modules for manipulating_
↳atoms,	analyzing simulations and visualization
Desmond	High-speed molecular dynamics simulations of biological systems
DL_POLY	General purpose classical molecular dynamics (MD) simulation
ESPResSo	Extensible Simulation Package for Research on Soft matter
GAMESS	General Atomic and Molecular Electronic Structure System (GAMESS) -
↳Ab initio	molecular quantum chemistry
babel	A chemical toolbox designed to speak the many languages of chemical_
↳data	
gpaw	A density-functional theory (DFT) Python code
gromacs	Perform molecular dynamics; simulate the Newtonian equations of_
↳motion for	systems with hundreds to millions of particles
nwchem	Methods for computing the properties of molecular and periodic systems
OpenMD	Open source molecular dynamics engine
Maestro	Schrödinger Maestro - a powerful, all-purpose molecular modelling_
↳envirotnment	

Compilers

GNU GCC	GNU Compiler Collection including front ends for C, C++, Objective-C_
↳and	Fortran
Cluster Studio	Intel Cluster Studio - High performance cluster tools to increase_
↳performance	and scalability
Open64	An open source, optimizing compiler for the Itanium and x86-64_
↳microprocessor	architectures
Oracle Java(TM)	Java Programing Language

Databases

hdf5	Data model, library, and file format for storing and managing data
------	--

Electronics

Octopus	A scientific program aimed at the ab initio virtual experimentation
---------	---

Engineering

ANSYS Workbench	Suite of advanced engineering simulation tools
Code_Saturne	Solve the Navier-Stokes equations for 2D, 2D-axisymmetric and 3D flows

Geography

GRASS GIS →suite	Free and open source Geographic Information System (GIS) software_
PROJ.4 →coordinates	Convert geographic longitude and latitude coordinates into cartesian_

Graphics and Imaging

DIL →simple syntax to load, save, convert, manipulate, filter and display a variety of_	Developer's Image Library - Cross-platform image library utilizing a_
POV-Ray	The Persistence of Vision Raytracer
vtk	Package for 3D graphics, modeling and image processing
Bsoft	Bernard's Software Package
CTFFIND3	CTF estimation
CTFFIND4	CTF estimation
Dynamo	Software environment for subtomogram averaging of cryo-EM data
EMAN2	Broadly based greyscale scientific image processing suite
FFmpeg →audio and video	A complete, cross-platform solution to record, convert and stream_
IHRSR++ →software	Extension of Iterative Helical Real Space Reconstruction (IHRSR)_
IMOD →reconstruction	Image processing, modeling and display programs for tomographic_
RELION	REgularised LIkelihood Optimisation
ResMap	Local Resolution Map Algorithm
SPIDER →fields	System for Processing Image Data from Electron microscopy and Related_

Languages

Anaconda Py2.7 →(python 2.7)	Completely free Python distribution including popular Python packages_
Anaconda Py3 →(python 3)	Completely free Python distribution including popular Python packages_
Cython	C-Extensions for Python
Glasgow →language Haskell	Haskell Compiler and interactive environment for the functional_
julia →technical computing	High-level, high-performance dynamic programming language for _
perl →years of development	A highly capable, feature-rich programming language with over 24_
php →suited for Web development	A widely-used general-purpose scripting language that is especially_
python	A remarkably powerful dynamic programming language
R	Language and environment for statistical computing and graphics
ruby →simplicity and productivity	A dynamic, open source programming language with a focus on_
IPython →Jupyter	Rich architecture for interactive computing, supporting Project_
Mono →cross platform applications	A software platform designed to allow developers to easily create_
Oracle Java(TM)	Java Programing Language

R	Language and environment for statistical computing and graphics
Scala	Multi-paradigm programming language built on top of the Java virtual_
↳machine.	
Virtualenv	Virtual Python Environment Builder

Libraries

GCC	GNU C/C++ Compiler
ANTLR	ANother Tool for Language Recognition - Language tool that provides a_
↳framework for	constructing interpreters, compilers, and translators
BLACS	Basic Linear Algebra Communication Subprograms - A linear algebra_
↳oriented message	passing interface
Caffe	A fast open framework for deep learning
CUDA Toolkit	Development environment for C and C++ developers building GPU-
↳accelerated applications	
CythonGSL	Cython interface for the GNU Scientific Library (GSL)
GEOS	Geometry Engine, Open Source
GDAL	Geospatial Data Abstraction Library
↳geospatial data formats	Translator library for raster_
GMP	Library for arbitrary precision arithmetic
GSL	GNU Scientific Library - A numerical library for C and C++ programmers
Rnetcdf	RNetCDF
libdc1394	C++ API to the PostgreSQL database management system.
Math-atlas	Automatically Tuned Linear Algebra Software - portably optimal linear_
↳algebra software	
mpi4py	Python bindings for the Message Passing Interface (MPI)
libgit2	Portable, pure C implementation of the Git core methods
boost	Free peer-reviewed portable C++ source libraries
fftw	C subroutine library for computing the discrete Fourier transform_
↳(DFT) in one or more dimensions	
fltk	A cross-platform C++ GUI toolkit providing modern GUI functionality_
↳without the bloat	
freeds	A set of libraries for Unix and Linux that allow programs to natively_
↳talk to Microsoft SQL Server and Sybase databases	
freetype	Freetype fonts package
graphicsmagick	Swiss army knife of image processing
imagemagick	An open source software suite for displaying, converting, and editing_
↳raster image files	
Blas-forum	Reference implementation for the C interface to the Legacy BLAS
blas	Routines that provide standard building blocks for performing basic_
↳vector and matrix operations	
clapack	LAPACK translated from Fortran to C
lapack	Linear Algebra PACKage - routines for equation solving systems
scalapack	A library of high-performance linear algebra routines for parallel_
↳distributed memory machines	
pil	Adds image processing capabilities to your Python interpreter
netcdf	NetCDF
zeroc	A modern distributed computing platform.
Img	Support for many image formats for Tk
JasPer	Reference implementation of the JPEG-2000 Part-1 standard
Lasagne	Lightweight library to build and train neural networks in Theano
libctl	Flexible control files for scientific simulations
libgdiplus	C-based implementation of the GDI+ API
Libxc	Libxc is a library of exchange-correlation functionals for density-
↳functional theory.	
LLVM Core	A modern source- and target-independent optimizer with code_
↳generation support	

```

matplotlib      2D plotting library for Python which produces publication quality_
↳figures
MPFR             C library for multiple-precision floating-point computations with_
↳correct rounding
NetCDF Fortran   Set of interfaces and libraries for array-oriented data access
NetCDF           Set of interfaces and libraries for array-oriented data access
numexpr          Fast numerical array expression evaluator for Python and NumPy
OpenBLAS         An optimized BLAS library
OpenCV           Open source computer vision and machine learning software library
OpenLibm         High quality, portable, standalone C mathematical library
pandas           Powerful data structures for data analysis, time series,and statistics
PCRE2            Perl Compatible Regular Expressions
Protocol Buff    Protocol Buffers are a way of encoding structured data in an_
↳efficient yet extensible format. Google uses Protocol Buffers for almost all of its_
↳internal RPC protocols and file formats.
pybedtools       Wrapper around BEDTools for bioinformatics work
pythonlevenshtein Python extension for computing string edit distances and_
↳similarities
PyGTK Libraries  GTK+ for Python
PyQt4            Python v2 and v3 bindings for Digia's Qt application framework
PyQt5            PyQt5 plots data with Numerical Python and PyQt
Pysam            Python module for reading and manipulating Samfiles
PyTables         Hierarchical datasets
Qt               A cross-platform application and UI framework
QuTiP            Quantum Toolbox in Python
Rmpi             Provides an interface (wrapper) to MPI APIs
ROOT             Set of OO frameworks to handle and analyze large amounts of data_
↳efficiently
RPy              A simple and efficient access to R from Python
Seaborn          Seaborn is a Python visualization library based on matplotlib.
SLICOT           Subroutine Library in Systems and Control Theory
Snappy Java      Snappy compressor/decompressor for Java
snpsites         Rapidly extract SNPs from a multi-FASTA alignment
SparseHash       An extremely memory-efficient hash_map implementation
SPIOL           Staden Package I/O Libraries - I/O libraries developed as part of the_
↳Staden Project
Theano           Define, optimize, and evaluate mathematical expressions involving_
↳multi-dimensional arrays efficiently
TBB              Threading Building Blocks - C++ template library that simplifies the_
↳development of software applications running in parallel
Trilinos         Algorithms for the solution of multi-physics engineering and_
↳scientific problems
UDUNITS          Programatic handling of units of physical quantities
Ceres Solver     C++ library for modeling and solving large complicated nonlinear_
↳least squares problems
GetPot           Tool to parse the command line and configuration files.
gflags           Library that implements commandline flags processing
glog             Application-level logging library
nanoflann        C++ header-only fork of FLANN, a library for KD-trees
pyBigWig         A python extension, written in C, for quick access to and creation of_
↳bigWig files.

```

Mathematics

```

eigen            C++ template library for linear algebra
Arpack-ng        Collection of Fortran77 subroutines designed to solve large scale_
↳eigenvalue problems

```

numpy	Fundamental package for scientific computing in Python
suitesparse	A suite of sparse matrix packages
octave	High-level interpreted language, primarily intended for numerical_
↪computations	
qhull	General dimension code for computing convex hulls
GCAL	Computational Geometry Algorithms Library
METIS	Serial Graph Partitioning and Fill-reducing Matrix Ordering
MGRIDGEN	Obtain a sequence of successive coarse grids that are well-suited for _
↪geometric multigrid methods	
grupdate	Fortran library for fast updates of QR and Cholesky decompositions
SciPy	Scientific tools for Python
SCOTCH	Graph and mesh/hypergraph partitioning, graph clustering, and sparse_
↪matrix ordering	
SMLT	The Shogun Machine Learning Toolbox - A large scale machine learning_
↪toolbox	

Medicine

FreeSurfer	A comprehensive library of analysis tools for FMRI, MRI and DTI brain_
↪imaging data	
FreeSurfer	An open source software suite for processing and analyzing (human)_
↪brain MRI images	

MPIs

mvapich2	MPI-2 over OpenFabrics-IB, OpenFabrics-iWARP, PSM, uDAPL and TCP/IP
mvapich	MPI-1 over OpenFabrics/Gen2, OprnFabrics/Gen2-UD, uDAPL, InfiniPath,_
↪VAPI and TCP/IP	
mpich2	A high-performance and widely portable implementation of the MPI_
↪standard (both MPI-1 and MPI-2)	
MPICH	A high-performance and widely portable implementation of the MPI_
↪standard (MPI-1, MPI-2 and MPI-3)	
Open MPI	A High Performance Message Passing Library

Physics

CASTEP	A leading code for calculating the properties of materials from first ↵
↵principles	
freEDA	Multi-physics simulator
Harminv	Extract mode frequencies from time -series data
openfoam	A C++ toolbox for the development of customized numerical solvers,↵
↵ and pre-/post-processing utilities	
LAMMPS	Molecular Dynamics Simulator - LAMMPS ('Large-scale Atomic/Molecular↵
↵Massively Parallel Simulator') is a molecular dynamics program from Sandia National↵	
↵Laboratories	
Meep	Finite-difference time-domain (FDTD) simulation software
MPB	MIT Photonic Bands - Electromagnetic eigenmode solver

Statistics

biogeme	Estimation of discrete choice models
fastlowess	An improved version of statsmodel's <code>lowess</code>
GGPlot	An implementation of the grammar of graphics <code>in</code> R

Tools

Bazel	Correct, reproducible, fast builds for everyone
bcrypt	Cross-platform file encryption utility
CMake	An extensible, open-source system that manages the build process in <code>an operating system and compiler-independent manner</code>
EC2 AMI Tools	Command-line tools to create and manage Amazon Machine Images
EC2 API Tools	Command-line tools for managing EC2 instances
Git	Git - the stupid content tracker
GNU Parallel	Shell tool for executing jobs in parallel using one or more computers
flex	The fast lexical analyzer
cpanminus	Get, unpack, build and install modules from CPAN
patchelf	Utility to modify the dynamic linker and RPATH of ELF executables
cmake	An extensible, open-source system that manages the build process in <code>an operating system and compiler-independent manner</code>
tau	Portable profiling and tracing toolkit for performance analysis of <code>parallel programs written in Fortran, C, C++, Java, Python</code>
OpenStackClient	Command-line client for OpenStack
pip	The PyPA recommended tool for installing and managing Python packages.
setuptools	Download, build, install, upgrade and uninstall Python packages <code>easyly!</code>
SIP	Automatically generate Python bindings for C and C++ libraries
h5utils	Utilities for visualization and conversion of scientific data in HDF5 <code>format</code>
mbuffer	Tool for buffering data streams

Visualization

Circos	A software package <code>for</code> visualizing data <code>and</code> information.
ants	Advanced Normalization Tools <code>for</code> brain <code>and</code> image mapping
OctoMap	A probabilistic, flexible, <code>and</code> compact 3D mapping library <code>for</code> robotic <code>systems</code>
ParaView	Data analysis <code>and</code> visualization

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 language 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, you can 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/e17/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 language 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 language 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 1 compute node
- AutoScaling disabled (disable with `alces configure autoscaling disable`)

Installing OpenFoam

The following section details how to install OpenFoam version 3.0.1 on your Alces Flight Compute environment. Note - access to the administrator user is required for this section.

You must first install the Gridware packages required to complete the following tutorial - install the required packages using the following example commands:

```
alces gridware install apps/openfoam
alces gridware install apps/paraview
```

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 `checkMesh` 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. Once all of the Mesh regions are selected, click the `Apply` button.
11. Click the `Play` button using the toolbar to run the output.

