On-demand
Research Compute Environments

Agenda

- Introductions
- Deploying your cluster
  - Requirements and configuration options
  - Deploying with CloudFormation templates
- Using your cluster
  - Your user environment and module files
  - Start and managing sessions
  - Applications
  - Using file and object storage
  - Job-scheduler; submitting and running jobs
- Terminating your cluster

Alces Flight

Overview

- Research compute environment; including:
  - Familiar Linux HPC cluster look and feel
  - HPC applications, libraries, MPIs and tools
  - Batch scheduler
  - Data management and accessibility tools
- Target audience
  - New and existing users
  - Scientific researchers and end-users
  - Ephemeral usage; rapid start and stop
- Unique selling points
  - Minimum possible setup time
  - Extremely customisable and expandable
  - Leverages benefits of public cloud

Flight on AWS

- Flight is an enabler for HPC on AWS:
  - Handles compute node start/stop efficiently, reducing idle compute cycles
  - Applications and data are centralised, minimising costs while cluster is idle
  - Simple scaling of clusters via batch-scheduler job submission
  - Ephemeral usage encourages users to manage their data better
  - Large library of available applications optimised for available instance types
  - Users can access from anywhere - no site requirements
  - Groups of users can collaborate on a single platform from any location
  - Standardised platform for training users new to AWS and HPC

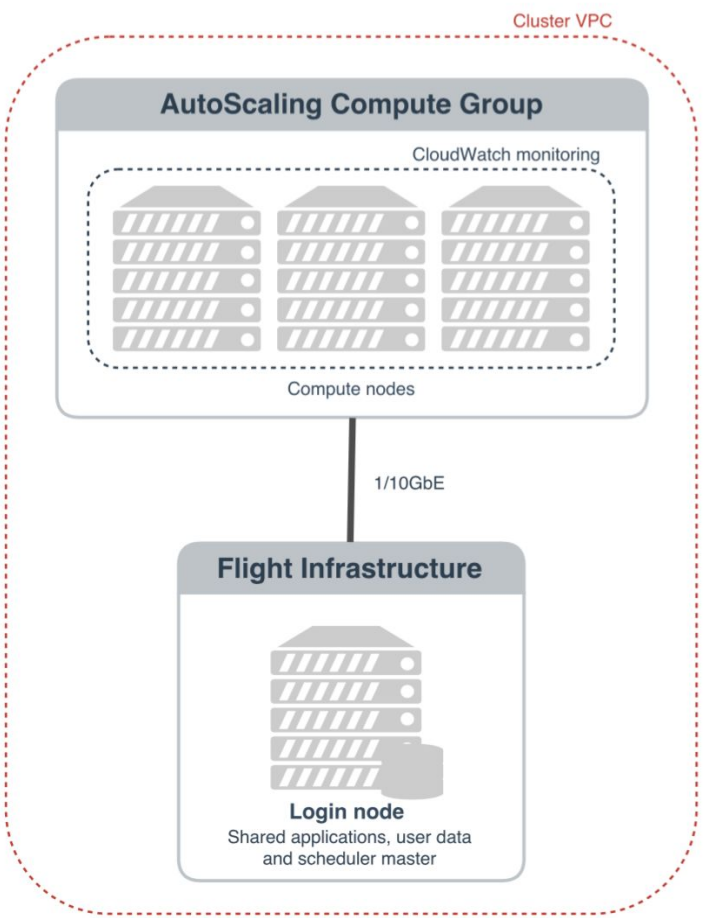
Flight Appliances

- Flight Compute
  - Launched from Marketplace: Single AMI with CloudFormation template
  - One login node + one or more compute nodes
  - Auto-configured shared NFS home directories
  - Batch job scheduler with example job-scripts
  - Access to more than 750 applications, + session and data management tools
- Additional support appliances available soon
  - Flight Access Manager
  - Flight Application Manager
  - Flight Storage Manager
  - Flight Galaxy appliance for BioInformatics pipelines

Deploying your cluster

Deploying your cluster

- Requirements for launching
  - *EU-West/Ireland region selected (pre-MP)*
  - SSH client and SSH keypair registered in AWS
  - AWS user account with IAM privileges
- Deployed architecture
  - Single cluster master node, hosting:
    - Cluster job scheduler,
    - Shared applications and module files
    - NFS area for user data
  - AutoScaling compute node group
    - Auto-detect local master and auto-configure
  - New VPC to contain the cluster



How do I deploy?

- Deployment using AWS CloudFormation
- Templates available to public and soon through AWS marketplace
  - https://s3-eu-west-1.amazonaws.com/flight-aws-marketplace/2016.1/8-node.json
  - https://s3-eu-west-1.amazonaws.com/flight-aws-marketplace/2016.1/16-node.json
  - https://s3-eu-west-1.amazonaws.com/flight-aws-marketplace/2016.1/32-node.json
- The templates create:
  - VPC, subnet, gateway, security group
  - A single cluster master node instance with EBS storage
  - Initial fixed number of compute nodes
    - Deployed as SPOT instances in an auto-scaling placement group
    - CloudWatch rule created along with IAM role to report data to AWS

Template options

- Three templates initially available in MarketPlace
- 8-node template
  - Initial 8 dedicated compute hosts using **c4.large** instance type - 2 cores and 4GB RAM
  - Recommended for evaluation and test usage
- 16-node template
  - Initial 16 dedicated compute hosts using **c3.4xlarge** instance type - 16 cores and 30GB RAM
  - Two SSDs per compute host for local scratch storage
  - Recommended for research use
- 32-node template
  - Initial 32 dedicated compute hosts using **c4.8xlarge** instance type - 36 cores and 60GB RAM
  - Recommended for research use

Deployment - steps

- From the EC2 management dashboard, navigate to the CloudFormation console
- Click **Create New Stack**
- Click **Specify an S3 template URL** and enter one of the previously specified template URLs
- Click **Next** to begin configuring your environment

Deployment

- With your own preferences, configure the following options
  - **Stack name**: Enter a name for your cluster, e.g. **hpc1**
  - **KeyPair**: Select your AWS keypair
  - **LoginType**: This defines the login node type - setting the number of cores and memory available to the login instance. T-instance types have an 8GB disk; C-instances types have 100GB.
  - **NetworkCIDR**: For increased security, limit access to the login node to this subnet
  - **SpotPrice**: Enter the price in USD per hour for the deployed spot instances, note - this is the maximum cost per hour you are prepared to pay
  - **SystemDiskSize**: Set the root disk size to use for each compute instance.
  - **Username:** Set the user login name that your AWS keypair will be associated with
- Click the **Next** button once you have verified your settings

Deployment

- From the **Options** page - click the **Next** button to review your settings
- From the **Review** page - click on the checkbox in the **Capabilities** section to allow creation of IAM role, then click the **Create** button to launch your cluster
- The cluster will begin creating its resources.
- Once the environment has finished creating, the public IP address of the cluster login node will be available from the **Outputs** tab

- Using your AWS key - SSH to the public IP address user you specified

Using the Compute Cluster

Example use-cases and workloads using your Alces Flight environment

Linux User Environment

Useful commands:

```
qhost
pwd
cd, mkdir, rmdir
cp, mv, rm
ssh ip-10-75-0-187
yum search <package>
sudo yum install <package>
```

- Your user account has full *sudo* access
  - Become root user as required
- Password-less SSH is automatically enabled between nodes
  - Use the **qhost** command to view cluster nodes
  - Log in to other nodes using SSH
- Shared storage for user data
  - Your home-directory is mounted on all nodes
  - Home directory is **/home/<username>/** or **~/**
    - e.g. `/users/alces/`
- Standard Linux environment
  - Install new packages on a single node using **yum**:
    - e.g. `sudo yum install nano`
  - Linux environment variables:
    - `$PATH`
    - `$LD_LIBRARY_PATH`
    - `$MANPATH`
    - `$USER`
    - `$PWD`

Working with multiple nodes

Useful commands:

`nodeattr`

`pdsh`

- Automatic node grouping support (genders)
  - Nodes are automatically added/removed
    - Default group is called **nodes**
  - Scales with compute cluster
  - Use **nodeattr** command to enumerate
    - e.g `nodeattr -s nodes`
- Parallel Distributed Shell (PDSH)
  - Use the **pdsh** command to run commands on multiple nodes
  - Manually specify nodes, or use genders support
  - Syntax:

```
pdsh [-g group,group,... | -w host,host,...] command
```

- Examples:

```
pdsh -g nodes uptime
pdsh -g nodes -x ip-10-75-0-123 'df -h /tmp'
pdsh -w ip-10-75-0-[123,125] 'tail -f /tmp/myfile.out'
pdsh -g nodes 'sudo yum -y install nano'
pdsh -g nodes -f 1 date
```

Graphical Desktop

Useful commands:

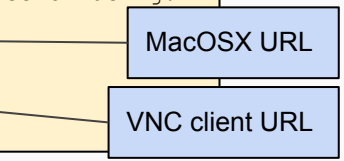
```
alces session avail
alces session enable <type>
alces session start <type>
alces session list
alces session info <id>
alces session kill <id>
xrandr
xrandr -s 1280x1024
```

- Graphical sessions can be launched on the login node
- Use the **alces session avail** command to list available types
- Use **alces session enable** command to enable a new type
  - e.g. `alces gridware enable chrome`
- Create a new session with **alces session start** command
  - e.g. `alces gridware start gnome`
- VNC session will be started with connection info printed:

```
[alces@login1(mycluster) ~]$ alces session start gnome
VNC server started:
       Identity: 5650cbf4-04c0-11e6-95a3-0af76f7bf051
       Type: gnome
       Host: 52.50.133.113
       Port: 5901
       Display: 1
       Password: u4NXvjpY

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

  vnc://alces:u4NXvjpY@52.50.133.133:5901
  52.50.133.133:5901
  52.50.133.133:1
```



- Use a VNC client to connect using the URL provided

Environment Modules

Useful commands:

```
module help
```

- Environment modules help users find and run applications, libraries and compilers
- Modules configure your shell variables
  - `$PATH, $LD_LIBRARY_PATH, $MANPATH`
  - For the user's current running session only (by default)
- Can be run at the command-prompt or in job-scripts
- New apps are installed with supporting module files
- Examples:

```
module avail
module load libs/gcc compilers/gcc
module list
module unload compilers/gcc
module whatis services/gridscheduler
module keyword gcc
module purge
module initadd apps/bowtie2
```

Cluster Software

Useful commands:

```
module avail
alces gridware list
```

- Alces Gridware project
  - Holds instructions for installing applications
  - Includes libraries, compilers and MPIs
  - Centrally manages applications on the login node
- Users can use gridware to
  - Download, compile and install software
  - Install depots of pre-built software
  - Prepare the environment to run jobs
- Applications are installed with a module file
- Use **module avail** command to list installed applications
- Example **alces gridware** commands:

```
alces gridware list
alces gridware search --names gromacs
alces gridware info apps/gromacs/4.6.1
```

Installing Applications

Useful commands:

```
alces gridware list
alces gridware install
   apps/bowtie2
module whatis apps/bowtie2
alces gridware purge
   apps/bowtie2
```

- New applications installed using **alces gridware install**
  - Specify the version required
  - Specify any variants (if required)

  e.g. `alces gridware install apps/bowtie2/2.2.6`
- Use **module avail** to view new module file once installed
- Use **module whatis apps/bowtie2** to view license
- Remove apps using **module purge apps/myapp**
- Multiple versions of applications can be installed
- Applications are available across all nodes
- Applications are installed into **local** depot

Gridware Depots

Useful commands:

```
alces gridware depot list
alces gridware depot fetch
alces gridware depot enable
module avail
```

- Applications can be collected together in depots
- Depots are pre-compiled for AWS
- Install a depot using **alces gridware depot fetch**

```
e.g. sudo su - -l -c 'alces gridware depot fetch
https://s3-eu-west-1.amazonaws.com/packages.alces-soft
ware.com/depots/demo'
```

- Use **alces gridware depot list** to view installed depots
- Use **alces gridware depot enable** to enable new depots for a user

```
e.g. alces gridware depot enable demo
```

- Use **module avail** to view new applications

Storing Data Files

Useful commands:

```
scp -i key <file> <remote>
sftp <file> <remotefile>
rsync -ravP -e "ssh -i key"
 <localfile> <remotefile>
```

- Users can copy files to the cluster using SCP
- Uses SSH keypair for passwordless transfer
- Data is encrypted in transit
- Many desktop clients available:
  - scp/sftp (Linux, Mac)
  - pscp/psftp (Windows)
  - WinSCP, FileZila (Windows explorer interface)
- Specify keypair, username and path to files:

```
e.g. scp -i aws.pub alces@52.50.133.133:/home/alces/file .
```

- Tools like **rsync** can make copying faster by skipping files that haven't changed:

```
e.g. rsync -ravP -e "ssh -i aws.pub" /storage/data
alces@52.50.133.133:/home/alces/.
```

Archiving Data

Useful commands:

```
alces storage enable
alces storage configure
alces storage avail
Alces storage use
alces storage get
alces storage put
```

- The **alces storage** command allows users to store files in S3, dropbox and POSIX services
- Use the **alces storage enable** command:
  - **s3** - Amazon S3 object storage
  - **dropbox** - Dropbox object storage
  - **posix** - An existing POSIX path
- Use the **alces storage configure** command to set up a new storage service

```
e.g. alces storage configure mys3 s3
```

- Select a service with **avail** and **use** commands
- Upload and download files with **get** and **put**

```
e.g. alces storage put localfile s3://remotefile
     alces storage --recursive get s3://jobs .
     alces storage mkbucket newbucket
```

Running an Application on the login node

Useful commands:

```
alces session start
xrandr
module load
```

- Batch jobs are submitted to run on compute nodes via the job scheduler
- Users can also run interactively on the login node
  - Example work-flow:

```
alces session start gnome
```

<connect to VNC desktop>

<from the desktop ***Application*** menu, select ***Terminal***>

```
module load apps/rstudio
rstudio
```

<inside Rstudio console>

```
demo(graphics)
```

Using the Job Scheduler

- Users submit jobs to a batch queue which are then run on compute nodes
- The job scheduler is responsible for:
  - Starting and stopping jobs
  - Assembling machinefile for parallel jobs
  - Storing printed job output in a file
- Users provide a **job script** which instructs the scheduler what to do to run their application
- Users can use **directives** to instruct the scheduler how to run their job; these may be
  - Included in the user's job-script
  - Provided on the command-line at submission time

Writing a job script

Useful commands:

```
qstat
qsub <job-script name>
cat <output file>
```

- A simple job-script:

```
#!/bin/bash -l
echo Hello from job $JOB_ID
sleep 60
echo I was run on node $HOSTNAME
```

- Submit the job using **qsub <jobscript>**
- View the job status using **qstat**
  - Output shows your unique job ID
  - Job status (r=running, qw=queuing/waiting)
- By default, job output stored in home-dir
  - Filename is **~/<jobscript-name>.o<job-ID>**

e.g. `cat ~/myjob.sh.o5`

Job scheduler directives

Useful commands:

```
qstat
qsub <job-script name>
cat <output file>
```

- The directive "**-o <filename>**" instructs the scheduler where to store job output
  - The directive can be added at submission time:

E.g. `qsub -o /home/alces/myoutput myjob.sh`

  - The directive can also be added in the job-script:

```
#!/bin/bash -l
#$ -o /home/alces/myoutput
echo Hello from job $JOB_ID
sleep 60
echo I was run on node $HOSTNAME
```

- Lines in a job-script starting with #$ are evaluated at submission time

Job scheduler directives

Useful commands:

```
qstat
qsub <directive>      <job-script name>
```

- More scheduler directives:

`-o <filename>`	Set job output file location
`-hold <job-ID,...>`	Wait until the specified job IDs have completed running
`-pe <PEname> <slots>`	Use named parallel environment, requesting a number of slots
`-p <priority>`	Set the job priority (0=normal, -1000 =low)
`-r <yes|no>`	Set the job to automatically rerun if terminated
`-l h_vmem=16G`	Request an amount of usable memory per slot (defaults to total / slots)
`-l rt=36:00:00`	Sets max job runtime to 36 hours

Example: Running an MPI job

Useful commands:

```
qsub imb.sh
```

- The **Intel Messaging Benchmark** (IMB) tests latency and bandwidth between compute nodes
- Write a simple job script that launches the IMB application, and submit to the cluster scheduler
- Watch the output to view results

```
#!/bin/bash -l
#$ -j y -N imb
#$ -o $HOME/imb.$JOB_ID.out
#$ -pe mpinodes-verbose 2 -cwd -V

module load apps/imb
mpirun -np 2 -npernode 1 IMB-MPI1
```

Template job-scripts

Useful commands:

```
alces template list
alces template copy <name>
  <new jobscript>
qsub <jobscript>
```

- A reference set of job-scripts are available using the **alces template** command
  - **alces template list** shows a list of available templates
    - **simple** - single-core serial job
    - **simple-array** - multiple single-core serial jobs
    - **smp** - multi-threaded job on a single node
    - **mpi-slots** - parallel job across multiple nodes, allocated by number of CPU cores
    - **mpi-nodes** - parallel job across multiple nodes, allocated by number of nodes
- The **alces template copy** command allows users to create new job scripts using a template

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

Managing Jobs

Useful commands:

```
qstat -j <job-ID>
qacct -j <job-ID>
qsub <job-script name>
cat <output file>
```

- Users can view the job queue using the **qstat** command; use **qstat -f** for full output
- Individual jobs can be deleted using **qdel <job-ID>**
- Job resource usage is shown using the **qacct <job-ID>** command (once job is completed)
- The status of compute nodes is shown using the **qhost** command

```
[alces-cluster@vlogin1(tiny) jobs]$ qhost
HOSTNAME                ARCH         NCPU LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
global                  -               -    -       -       -       -       -
vnode01                 linux-x64      20 0.01   58.8G  620.8M  32.0M     0.0
vnode02                 linux-x64      20 0.02   58.8G  638.0M  32.0M     0.0
vnode03                 linux-x64      20 0.01   58.8G  651.0M  32.0M     0.0
vnode04                 linux-x64      20 0.01   58.8G  622.5M  32.0M     0.0
vnode05                 linux-x64      20 0.01   58.8G  682.0M  32.0M     0.0
vnode06                 linux-x64      20 0.01   58.8G  642.6M  32.0M     0.0
vnode07                 linux-x64      20 0.01   58.8G  667.6M  32.0M     0.0
vnode08                 linux-x64      20 0.01   58.8G  642.5M  32.0M     0.0
vnode09                 linux-x64      20 0.01   58.8G  661.3M  32.0M     0.0
```

Terminating your cluster

- Before termination, store any job-scripts, data and output files created
  - Use **alces storage put --recursive** command to check-in data you want to keep
- List any applications you've installed
  - Use **module list** to view applications installed
  - Use **alces gridware depot export** to save your applications as a TAR file
- Delete the cluster stack from the *AWS CloudFormation* console
- Once deleted, users are no longer billed for the instances or software

Start computing today

- Many applications to explore
- Lots more help available online
  - http://alces-flight-appliance-docs.readthedocs.org/en/latest/

Thank you

www.alces-flight.com
http://alces-flight.github.io