IS Research Services — 2019-20

Introduction to Eddie

Schedule

- 1 **Introduction**
- 2 break
- 3 **Practical session 1**
- 4 Lunch
- 5 **Practical session 2**
- 6 break
- 7 **Practical session 3**
- 8 End

If you've not already done so:

- Please connect to eddie using your AD credentials
- Use mobaxterm or putty if using Windows
- Use ssh from a terminal if using a Mac or Linux machine

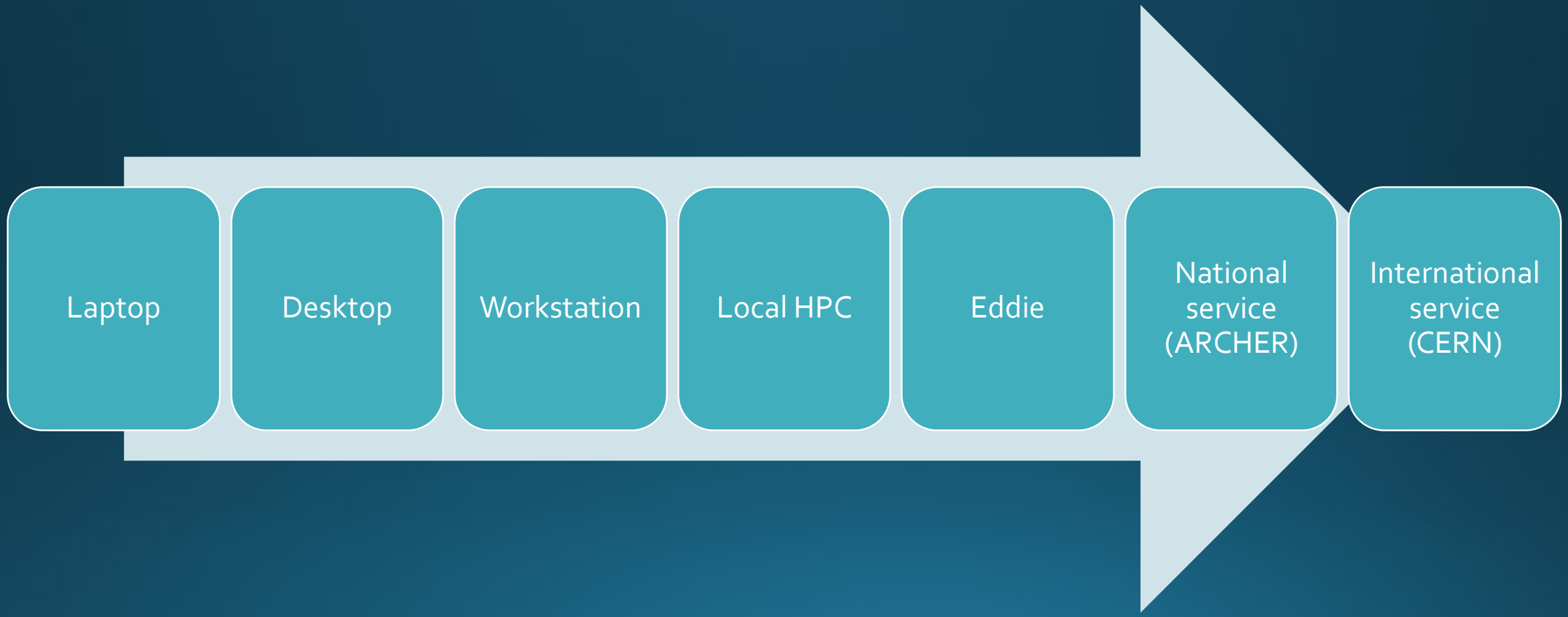
Eddie 3

- 1st and 2nd generations run on behalf of College of Science and Engineering
- **3rd generation: joint procurement between The University, IGMM, Roslin Institute and others**
- Accessible by **any** researcher in The University
- ~400 nodes, ~8000 cores, ~100 TB total RAM
- Eddie uses the Linux operating system (Scientific Linux 7.x)

Linux

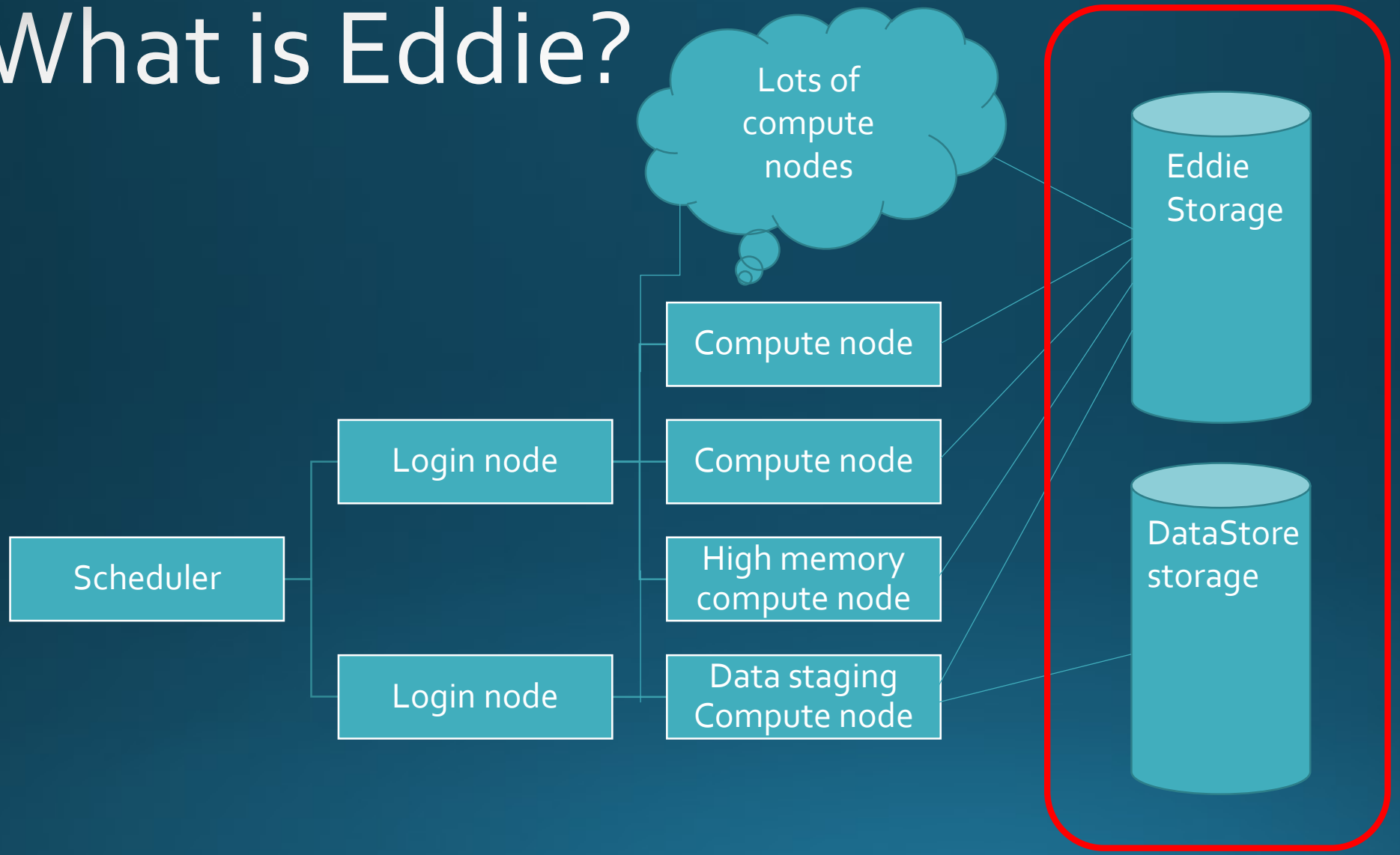
- Linux is a *highly recommended* pre-requisite for using Eddie
- Software Carpentry provides an online Linux course: http://swcarpentry.github.io/shell-novice/

Why use Eddie?



Eddie Hardware

What is Eddie?

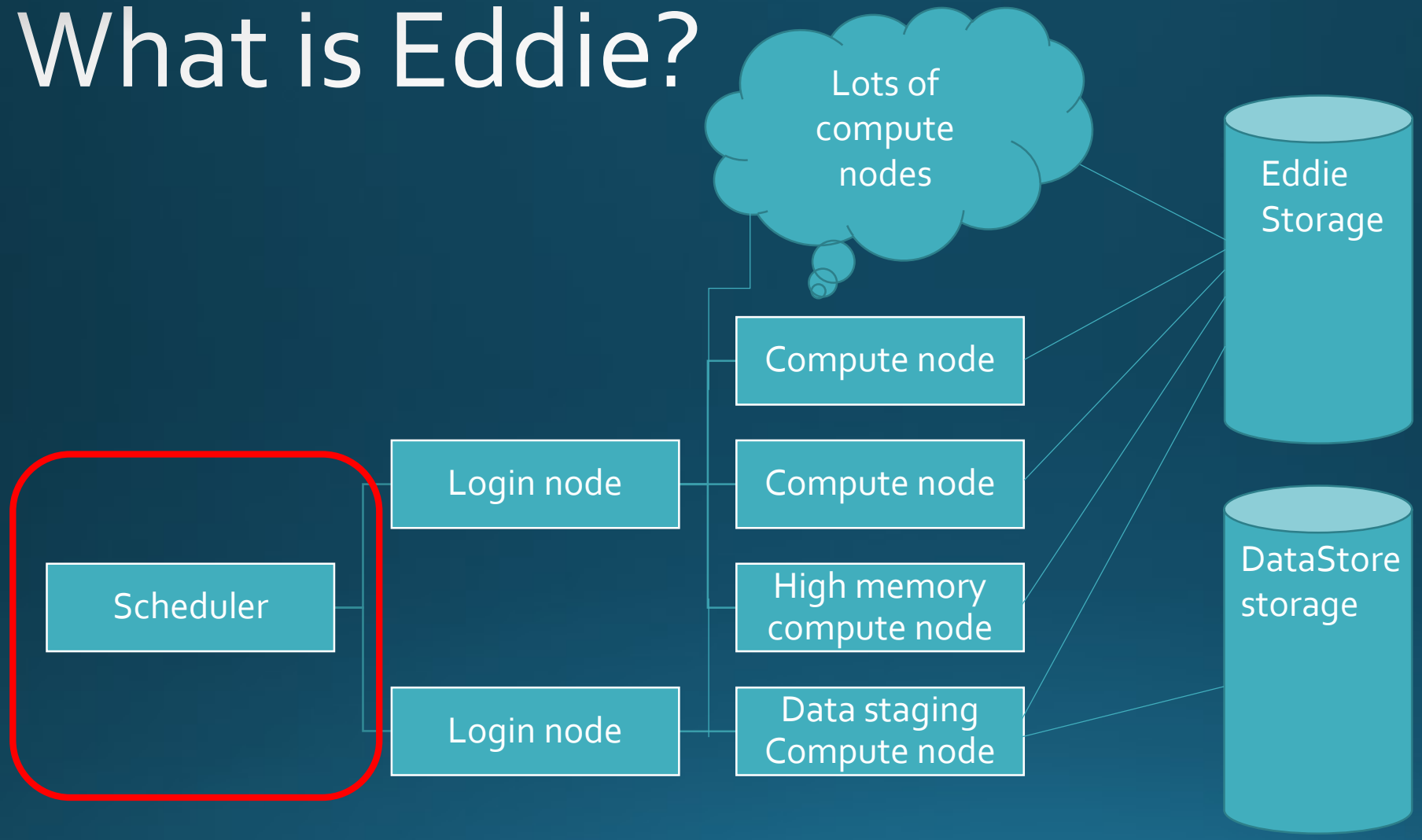


Filesystems

- **Eddie** (2.5 PB)
  - GPFS (now IBM Spectrum Scale) on high performance spinning disk + SSD — ***fast, expensive***
  - `/exports/<college*>/eddie/<group name>`
  - `/exports/eddie/scratch/<UUN>` (Files deleted after 28 days)
- **DataStore** (> 12 PB)
  - Commodity platform, highly resilient — ***slow, cheap***
  - Only accessible from **staging** and most **interactive** nodes
  - Need to copy (**stage**) data from DataStore to Eddie first
  - `/exports/<college*>/datastore/<group name>`

`*college` is one of `csce`, `cmvm`, `chss` or `sg`

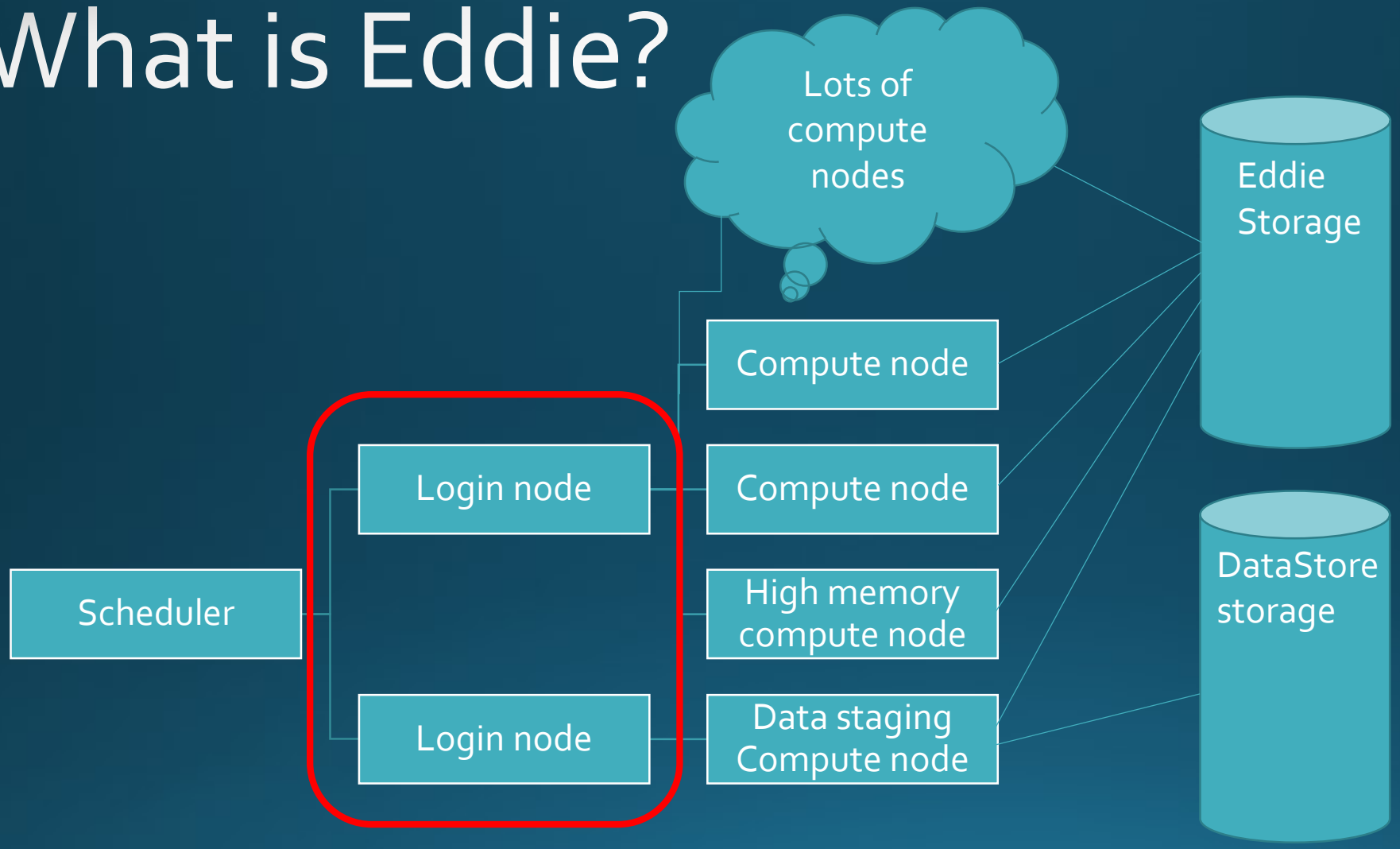
What is Eddie?



The Scheduler

- Controls a “Batch System”
  - Execution of a series of programs (**jobs**) *without* manual intervention
  - Users have access to a few nodes for *interactive* work
- The Scheduler will:
  - Identify each job’s resource requirements
  - Schedule jobs for execution when resources are available
  - Manage running jobs
  - Log usage and other information about each job

What is Eddie?



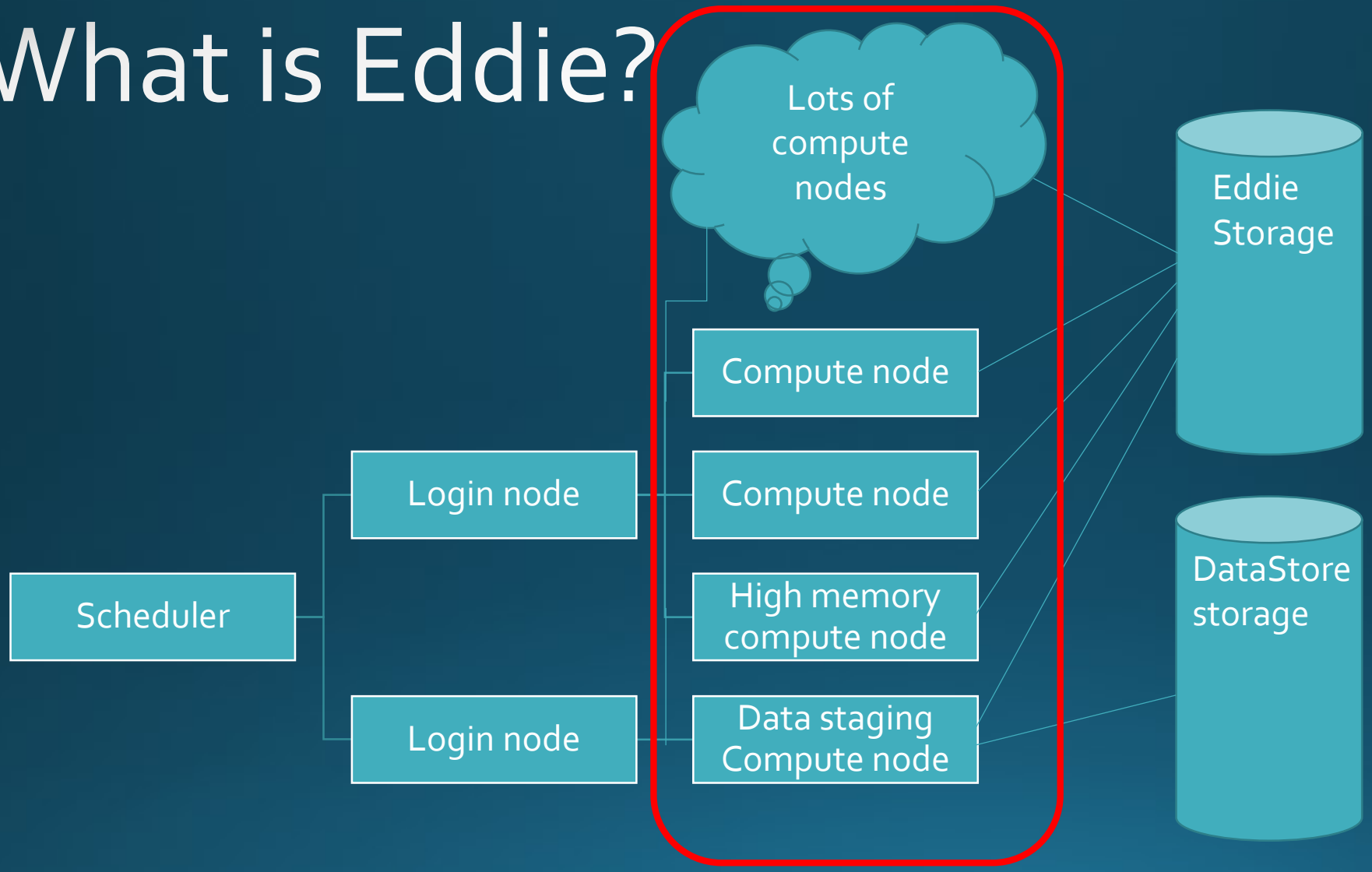
Login nodes

- Accessed by everybody
- Multiple hosts
- Only 16 CPUs, 64Gb RAM
- Used for low-intensity tasks
- (ie, **not** production jobs)
- Editing files, moving data, monitoring and submitting jobs, very short tests are all fine

Command line tools

- `qsub` submit a job to the queue
- `qstat` examine a job's current status
- `qlogin` start an interactive compute session
- `qacct` get a job's accounting information
- `qdel` delete jobs from the queue
- `qalter` change the attributes of (pending) jobs
- + others

What is Eddie?



Compute nodes

- Multiple types
- Jobs are allocated based on requested resources
- Some nodes have 3Tb RAM (most have 64Gb)
- Some nodes have 48 cores (many only have 8 or 16)
- Some nodes have 4 week queues (most are 48hr)
- Some nodes have GPUs (only a few)
- Staging nodes can access datastore
- Interactive nodes can be really useful, especially with GUIs

Jobs

- A **job** is specified by a job script
- Two parts:
  1. How to schedule the job — what resources are required (CPU cores, memory, time, etc.)
  2. What the job does — the application

- The job is submitted to the cluster with the qsub command:
  - `qsub job_script.sh`

Example job script

```
#!/bin/bash

# Grid Engine options go here
#$ -cwd
#$ -l h_rt=48:00:00

# Job payload
python -u myapplication.py
```

Job specification

- Most important parameters
  - Number of CPU cores (default 1)
  - Maximum memory (default 1 GB)
  - Length of job (default 48 hours)
- Maximum job length
  - Generally: 28 days
  - Many more nodes accept jobs with a 48 hour maximum
- Available memory
  - Node memory from 64 GB to 3 TB
  - Fewer nodes with large memory
- Estimating memory and runtime can be a challenge

Essential job options

- Working directory:
  - `-cwd`
  - `-wd /path/to/directory`
- Runtime:
  - `-l h_rt=48:00:00`
- Memory:
  - `-l h_vmem=4G`
  - (memory is specified per-core)
- Cores:
  - `-pe sharedmem 4`
- See man page for more options (`man qsub`)

Job status

- The `qstat` command shows the status of jobs

`qstat`	Show your jobs
`qstat -u '*'`	Show jobs for all users
`qstat -j <job-ID>`	Detailed information on pending/running jobs, including reasons for a job being rejected

Interactive Jobs

- Applications that require an interactive session
- Use the `qlogin` command

- Takes many of the options that `qsub` accepts, e.g.:

```
qlogin -l h_vmem=8G
```

qdel

- Remove job(s) from the queue
- Examples:

`qdel <job-ID>`	Delete job from queue
`qdel -f <job-ID>`	Force deletion

Where should I work?

Login nodes

- No intensive computations (CPU and memory limited)
- Preparing, submitting and monitoring jobs
- Small bookkeeping activities (moving files, examining job output)

Interactive jobs (`qlogin`)

- Applications that require an interactive session
- Job development and testing
- Sessions may not be available (nodes already full)
- Maximum 48 hours

Batch jobs (`qsub`)

- Applications that can be run unattended
- Recommended option
- Mostly 48 hour maximum, some nodes 28 days

Practical Session 1

1. Login to Eddie
2. Run `getcoursefiles eddie` to get the course materials to `$HOME/IS_Courses/eddie`
3. All exercises are at:

   https://www.wiki.ed.ac.uk/display/ResearchServices/Introduction+to+Eddie+Course+-+Exercises

   Or https://edin.ac/2NhHIZ4
4. Try Exercises 1, 2 and 3

Data Staging

- Transferring data between DataStore and Eddie
- Use the staging queue: `#$ -q staging`
- Interactive: `qlogin -q staging`
- Sample staging job provided
- Often used with job dependencies

Data Staging — example job

```
#!/bin/bash

# Choose the staging environment
#$ -q staging

# Source path on DataStore
SOURCE=/exports/csce/datastore/<path of directory to copy>

# Destination path on Eddie
DESTINATION=/exports/eddie/scratch/$USER/

# Do the copy with rsync (avoid -p or -a options)
rsync -rl ${SOURCE} ${DESTINATION}
```

Software/Applications

- Eddie uses the *environment modules* tool to manage applications
- Examples:

`module avail`	List available software applications
`module load <application>/<version>`	Make application available (in order to use it)
`module unload <application>/<version>`	Unload the application
`module list`	List already loaded applications
`module whatis <application>/<version>`	Show info about an application and version
`module help <application>/<version>`	Show help about an application and version

Modules in job scripts

```
#!/bin/bash

# Grid Engine options go here:
#$ -cwd -l h_rt=00:10:00,h_vmem=2G

# Setup the environment modules command
source /etc/profile.d/modules.sh

# Load modules if required
module load anaconda/5.0.1

# Job payload
python -u myapplication.py
```

Graphical Applications

- Can be run using *X11 forwarding*
- Application runs on Eddie, the display is sent over the network (forwarded) to your desktop/laptop

- Windows: MobaXterm does this automatically
- Mac: `ssh -Y uun@eddie3.ecdf.ed.ac.uk` (needs XQuartz)
- Linux: `ssh -X uun@eddie3.ecdf.ed.ac.uk`

Graphical Applications

1. Connect to Eddie (MobaXterm/ssh)
2. Create an interactive session e.g.:

```
qlogin -pe interactivemem 4 -l h_vmem=4G
```

3. Load application with `module load`
4. Run graphical application

(`xclock` is a useful application for debugging X11 forwarding)

Array Jobs

- A number of (mostly) identical tasks
- Suitable for:
  - analysing many files in an identical way
  - parameter sweeps
  - …
- Option: `-t`, e.g.:

```
-t 1-100
```

Tasks are identified by environment variable `SGE_TASK_ID`,
In this example, `$SGE_TASK_ID` takes the values 1 to 100

Practical Session 2

Try Exercises 4, 5 and 6

Parallel environments

Requesting **more than one CPU core**

```
-pe <name> <no. of cores>
```

`sharedmem`

- Multiple cores on one node
- Suitable for shared-memory programming e.g. OpenMP

`interactivemem`

- Same as `sharedmem`, but for interactive jobs

Parallel environments

To request an interactive session with 2 CPU cores and 8 GB (2x4 GB) of RAM:

```
qlogin -pe interactivemem 2 -l h_vem=4G
```

More Parallel environments

`mpi`

- Multiple (whole) nodes — 16-core nodes
- Suitable applications that support the MPI standard

`gpu`

- Requesting GPUs — specifies the number of GPUs

`scatter`

- Any number of cores wherever they are free

GPUs — What's available?

GPU	Quantity* (free)	RAM	TFLOPS (double)	TFLOPS (single)
TITAN X (Pascal)	80 (20)	12 GB (uncorrected)	0.3	10.2–11.0
K80 (dual Tesla)	42 (24)	2 x 12 GB (corrected)	1.8–2.9	5.6–8.7

* approximate numbers

See: https://www.wiki.ed.ac.uk/display/ResearchServices/GPUs

Requesting GPUs

- Use the GPU parallel environment (`-pe gpu`), for example:

```
qsub -pe gpu 1 job_script.sh
qsub -pe gpu-titanx 1 job_script.sh
```

- All GPU nodes support interactive logins:

```
qlogin -pe gpu 1 -l h_vmem=8G
qlogin -pe gpu-titanx 1 -l h_vmem=8G
```

- *Interactive* jobs need to set `CUDA_VISIBLE_DEVICES`:

```
$ source
/exports/applications/support/set_cuda_visible_devices.sh
```

Deep learning libraries

- *Tensorflow*, *PyTorch*, *Theano*, *Keras*, etc.
- Recommended to install using Anaconda
- Can be memory 'unfriendly'
- Need to use a compatible CUDA version (currently 7.5, 8.0)
- See: https://www.wiki.ed.ac.uk/display/ResearchServices/TensorFlow

Job Failures

- Your jobs **will** fail
- Some reasons:
  - Syntax error in job script
  - Job can’t start (directory paths wrong; authentication problems)
  - Job runs out of memory or time
  - Application error
  - …

Resource estimation

- Need to estimate the following:
  - Number of CPU cores (default 1)
  - Maximum memory (default 1 GB)
  - Length of job (default 48 hours)
- How?
  - Submit test jobs with e.g. smaller datasets
  - Monitor with `qstat`, `qacct` or `job_summary.sh`
  - Scale-up values to actual dataset size

qacct

- Extract information from accounting file
  (See `man accounting` for full details of output)

- Examples:

  `qacct -j <job-ID>` shows details of the job

  `qacct -o <username>` shows total usage by that user

- `job_summary.sh` also provides a summary of this information

Practical Session 3

Try Exercises 7, 8 and 9

Documentation

- https://www.wiki.ed.ac.uk/display/ResearchServices
- http://edin.ac/2nuZMUG