BINF2111 – Introduction to Bioinformatics Computing

Research Cluster – Cluster up

Richard Allen White III, PhD
RAW Lab

Lecture 13 – Tuesday Oct 1st, 2024

Learning Objectives

- Review quiz/bonus

- Github terminal authentication

- Github functions

- Cluster set-up

- Run on the cluster

- Quiz 13

Bonus 11

- Write a bash script that converts this into a csv file (specific files and all files)?

With this file:
Rat,steven,bear,Xi
Olf,thor,flower,Ton
Lazy,aaron,larry,jose
old,thor,flower,Ton

Bonus 12

- Write a bash script that converts this into a csv file (**specific files** and all files)?

Bonus 12

- Write a bash script that converts this into a csv file (**specific files** and all files)?

```
1 #!/bin/bash/
2
3 input=$1
4
5 sed 's/,/\t/g' $input
6
7 function print_to_terminal(){
8     echo "Your comma-separated file has been converted to tab-delim file" >$(tty)
9     echo -n "Wise choice!" >$(tty)
10 }
11
12 output=$(print_to_terminal)
13
```

Bonus 12

- Write a bash script that converts this into a csv file (specific files and **all files**)?

Bonus 12

- Write a bash script that converts this into a csv file (specific files and **all files**)?

```
#!/bin/bash/

for i in *csv;
do
    sed 's/,/\t/g' "$i" >$(basename "$i" .csv).tsv
done

function print_to_terminal(){
    echo "Your comma-separated file has been converted to tab-delim file" >$(tty)
    echo -n "Wise choice!" >$(tty)
}

output=$(print_to_terminal)
```

Question

Write a bash script that prints any range of lines you give it from a file.

For example, bash script.sh file.tsv (prints lines 2 to 5). Place code here and make an executable script.

HOW WOULD YOU DO THIS?

Question

Write a bash script that prints any range of lines you give it from a file.

For example, bash script.sh file.tsv (prints lines 2 to 5). Place code here and make an executable script.

HOW WOULD YOU DO THIS?

```
1 #!bin/bash
2
3 #variables
4 file=$1
5 range1=$2
6 range2=$3
7
8 #range finder
9 sed -n $2,$3"p" $file
```

```
1 #!/bin/bash
2
3 num1=$1
4 num2=$2
5 file=$3
6
7
8 sed -n "$1,$2p" $3
```

Question

Write a bash script that prints any range of lines you give it from a file.

For example, bash script.sh file.tsv (prints lines 2 to 5). Place code here and make an executable script.

HOW WOULD YOU DO THIS?

```
#!bin/bash

#variables
file=$1
range1=$2
range2=$3

#range finder
sed -n $2,$3"p" $file
```

```
#!/bin/bash

num1=$1
num2=$2
file=$3


sed -n "$1,$2p" $3
```

HOW WOULD I RUN THESE?

Question

Write a bash script that prints any range of lines you give it from a file.

For example, bash script.sh file.tsv (prints lines 2 to 5). Place code here and make an executable script.

HOW WOULD YOU DO THIS?

```
1 #!bin/bash
2
3 #variables
4 file=$1
5 range1=$2
6 range2=$3
7
8 #range finder
9 sed -n $2,$3"p" $file
```

bash script.sh file.tsv 2 5

```
1 #!/bin/bash
2
3 num1=$1
4 num2=$2
5 file=$3
6
7
8 sed -n "$1,$2p" $3
```

bash script.sh 2 5 file.tsv

Question

Write a bash script that prints any range of lines you give it from a file.

For example, bash script.sh file.tsv (prints lines 2 to 5). Place code here and make an executable script.

HOW WOULD YOU DO THIS? With user inputs?

Question

Write a bash script that prints any range of lines you give it from a file.

For example, bash script.sh file.tsv (prints lines 2 to 5). Place code here and make an executable script.

HOW WOULD YOU DO THIS? With user inputs?

```
1 #!/bin/bash
2
3 echo "Enter first number"
4 read num1
5
6 echo "Enter second number"
7 read num2
8
9 echo "Enter file name"
10 read file1
11
12 cat --number $file1 | head -$num2 | tail +$num1
```

Question

Oh NO! You tried to save your file from a text editor and it got corrupted (corrupted.fastq).You must now find where the file is corrupted by looking at what is missing in the file?

Write a command or script (better) to find the corrupted lines then print their line numbers.

HOW WOULD YOU DO THIS?

Another way?

Question

Oh NO! You tried to save your file from a text editor and it got corrupted (corrupted.fastq).You must now find where the file is corrupted by looking at what is missing in the file?

Write a command or script (better) to find the corrupted lines then print their line numbers.

HOW WOULD YOU DO THIS?

```
1 #!/bin/bash
2
3 var1=$1
4 var2=$2
5
6 diff -u $1 $2
```

Another way?

Question

Oh NO! You tried to save your file from a text editor and it got corrupted (corrupted.fastq).You must now find where the file is corrupted by looking at what is missing in the file?

Write a command or script (better) to find the corrupted lines then print their line numbers.

HOW WOULD YOU DO THIS?

```
1 #!/bin/bash
2
3 diff $1 $2 --color
```

Another way?

Question

Write a bash script or command to find sequences with >4 N’s in the sequence (MultiN.fastq), have it print the header of the sequence and the sequence itself.

Question

Write a bash script or command to find sequences with >4 N's in the sequence (MultiN.fastq), have it print the header of the sequence and the sequence itself.

```
1 #!/bin/bash
2
3 input=$1
4
5 grep -B1 '\(.*N\)\{4\}' $1
```

Question

Write a bash script or command to find sequences with >4 N’s in the sequence (MultiN.fastq), have it print the header of the sequence and the sequence itself.

Question

Write a bash script or command to find sequences with >4 N's in the sequence (MultiN.fastq), have it print the header of the sequence and the sequence itself.

```
1  #!/bin/bash
2
3  input=$1
4
5  N_detector(){
6      grep -B 1 "NNNN" $input
7  }
8
9  output=$(N_detector)
10
11 echo $output
```

Lab 5 Bonus

```
#!/bin/bash

input=$1

#Count start codons
count_starts(){
    grep "^ATG" $input | wc -l
}

number_starts=$(count_starts)

#Count stop codons
count_stops(){
    egrep "TAA$|TAG$|TGA$" $input | wc -l
}

number_stops=$(count_stops)

#Count codons for arginine
count_arg(){
    egrep "CGT|CGC|CGA|CGG|AGA|AGG" $input | wc -l
}

number_arg=$(count_arg)

#Count codons for serine
count_ser(){
    egrep "AGT|AGC|TCT|TCC|TCA|TCG" $input | wc -l
}

number_ser=$(count_ser)

#final table
echo -n "Number of starts: "
echo $number_starts
echo -n "Number of stops: "
echo $number_stops
echo -n "Number of Arginine: "
echo $number_arg
echo -n "Number of Serine: "
echo $number_ser
```

```

#Count codons for arginine
count_arg(){
    egrep "CGT|CGC|CGA|CGG|AGA|AGG" $input | wc -l
}

number_arg=$(count_arg)

#Count codons for serine
count_ser(){
    egrep "AGT|AGC|TCT|TCC|TCA|TCG" $input | wc -l
}

number_ser=$(count_ser)

#final table
echo -n "Number of starts: "
echo $number_starts
echo -n "Number of stops: "
echo $number_stops
echo -n "Number of Arginine: "
echo $number_arg
echo -n "Number of Serine: "
echo $number_ser

#Convert codons
convert_cod(){
    sed 's/^ATG/M/g' $input |
    sed 's/TAA$/*/g' |
    sed 's/TAG$/*/g' |
    sed 's/TGA$/*/g' |
    sed 's/CGT/S/g' |
    sed 's/CGC/S/g' |
    sed 's/CGA/S/g' |
    sed 's/CGG/S/g' |
    sed 's/AGA/S/g' |
    sed 's/AGG/S/g' |
    sed 's/CGT/R/g' |
    sed 's/CGC/R/g' |
    sed 's/CGA/R/g' |
    sed 's/CGG/R/g' |
    sed 's/AGA/R/g' |
    sed 's/AGG/R/g'
}
convert=$(convert_cod)

echo "Converting codons to listed amino acids: "
echo $convert
```

Lab 5 Bonus

```bash
#!/bin/bash

#Write a bash script that count the number of ATG (starts), Serine (S), Arginine (R), and TAA, TAG, TGA (stops)
#from the example2.fasta file then converts them into amino acid M, S, R, and * for TAA, TAG, TGA for stop codon.
#Remember that ATG encodes for methonine so the only count the from the beginning of the sequence or the end for the stops.
#Serine and Arginine can be throughout the sequence.

input=$1

#Methionine ATG
echo "Total amount of Methionine (START)"
grep -o "^ATG" $1 | wc -l

#Serine AGA AGG AGT AGC TCA TCG
echo "Total amount of Serine"
grep -Eo "AGA|AGG|AGT|AGC|TCA|TCG" $1 | wc -l
echo "Amount of AGA"
grep -Eo "AGA" $1 | wc -l
echo "Amount of AGG"
grep -Eo "AGG" $1 | wc -l
echo "Amount of AGT"
grep -Eo "AGT" $1 | wc -l
echo "Amount of AGC"
grep -Eo "AGC" $1 | wc -l
echo "Amount of TCA"
grep -Eo "TCA" $1 | wc -l
echo "Amount of TCG"
grep -Eo "TCG" $1 | wc -l

#Arginine GCA GCG GCT GCC TCT TCC
echo "Total amount of Arginine"
grep -Eo "GCA|GCG|GCT|GCC|TCT|TCC" $1 | wc -l
echo "Amount of GCA"
grep -Eo "GCA" $1 | wc -l
echo "Amount of GCG"
grep -Eo "GCG" $1 | wc -l
echo "Amount of GCT"
grep -Eo "GCT" $1 | wc -l
echo "Amount of GCC"
grep -Eo "GCC" $1 | wc -l
echo "Amount of TCT"
grep -Eo "TCT" $1 | wc -l
echo "Amount of TCC"
grep -Eo "TCT" $1 | wc -l

#Arginine GCA GCG GCT GCC TCT TCC
echo "Total amount of Arginine"
grep -Eo "GCA|GCG|GCT|GCC|TCT|TCC" $1 | wc -l
echo "Amount of GCA"
grep -Eo "GCA" $1 | wc -l
echo "Amount of GCG"
grep -Eo "GCG" $1 | wc -l
echo "Amount of GCT"
grep -Eo "GCT" $1 | wc -l
echo "Amount of GCC"
grep -Eo "GCC" $1 | wc -l
echo "Amount of TCT"
grep -Eo "TCT" $1 | wc -l
echo "Amount of TCC"
grep -Eo "TCT" $1 | wc -l

#STOP TAA TAG TGA
echo "Total amount of STOP"
grep -Eo "TAA$|TAG$|TGA$" $1 | wc -l
echo "Amount of TAA"
grep -Eo "TAA$" $1 | wc -l
echo "Amount of TAG"
grep -Eo "TAG$" $1 | wc -l
echo "Amount of TGA"
grep -Eo "TGA$" $1 | wc -l

#seds
sed "s/^ATG/M/g" $1 |
sed "s/AGA/S/g" |
sed "s/AGG/S/g" |
sed "s/AGT/S/g" |
sed "s/AGC/S/g" |
sed "s/TCA/S/g" |
sed "s/TCG/S/g" |
sed "s/GCA/A/g" |
sed "s/GCG/A/g" |
sed "s/GCT/A/g" |
sed "s/GCC/A/g" |
sed "s/TCT/A/g" |
sed "s/TCC/A/g" |
sed "s/TAA$/*/g" |
sed "s/TAG$/*/g" |
sed "s/TGA$/*/g"
```

Question

Write a bash script that converts this into a tsv file (**specific files**)?

Question

Write a bash script that converts this into a tsv file (**specific files**)?

```
#!/bin/bash
# Write a bash script that converts this into a tsv file (specific files and all files)?

file1=$1
file2=$2

cat $1 | tr ',' '\t'  > $2

```

```
#!/bin/bash/

input=$1

sed 's/,/\t/g' $input

function print_to_terminal(){
    echo "Your comma-separated file has been converted to tab-delim file" >$(tty)
    echo -n "Wise choice!" >$(tty)
}

output=$(print_to_terminal)
```

Question

Write a bash script that converts this into a tsv file (**specific files as a function**)?

Question

Write a bash script that converts this into a tsv file (**specific files as a function**)?

```
1 #!/bin/bash
2
3 #Load Variables
4 input=$1
5
6 function csv_tsv_convert(){
7     sed 's/,/\t/g' $input
8 }
9
10 output=$(csv_tsv_convert)
11
12 echo $output
```

Question

Write a bash script that converts this into a tsv file (specific files and **all files**)?

Question

Write a bash script that converts this into a tsv file (specific files and **all files**)?

```
#!/bin/bash/

for i in *csv;
do
    sed 's/,/\t/g' "$i" >$(basename "$i" .csv).tsv
done

function print_to_terminal(){
    echo "Your comma-separated file has been converted to tab-delim file" >$(tty)
    echo -n "Wise choice!" >$(tty)
}

output=$(print_to_terminal)
```

Question

Print the lengths all the sequences in a fastq file

```
1 #!/bin/bash
2
3 input=$1
4
5 awk 'NR%4 == 2 {lengths[length($0)]++} END {for (l in lengths) {print l, lengths[l]}}' $1
```

bioawk -c fastx '{ print $name, length($seq) }' ~/UNCC/BINF2111/data/MultiN.fastq

```
1 #!/bin/bash
2
3 input=$1
4
5 cat $input | awk '{if(NR%4==2) print length($1)}' | sort -n | uniq -c
```

Github – Basic Functions

git clone: Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository

git status: displays the state of the working directory and the staging area.

git pull: update the local version of a repository from a remote. By default, git pull does two things. Updates the current local working branch (currently checked out branch) Updates the remote tracking branches for all other branches

git commit: Create a new commit containing the current contents of the index and the given log message describing the changes.

git add: Updates the index using the current content found in the working tree, to prepare the content staged for the next commit.

git fetch: checks server for updates without ‘pulling’ them

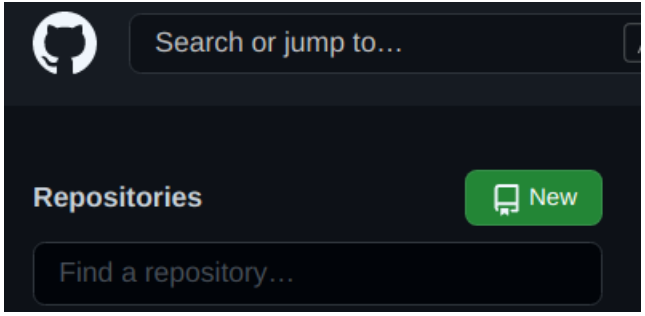
Github – Basic Functions

- Pulling data from github
  - git clone https://github.com/....
  - Git pull (for updating repo after someone else makes changes)
- Git needs to authenticate before you can push, or pull from a private repo.
- Push to github (assuming you are in root directory of the code)
  - git add .
  - git commit –m “Description of changes”
  - git push

Github – Basic Functions

- Show changes
  - git status
  - git fetch (checks server for updates without ‘pulling’ them)
- Create a new branch (for working on fixes/features)
  - git checkout –b <branch name> (i.e. dev)
- Switch between branches
  - git checkout <name>
- List local and remote branches
  - git branch -a

Github – Practice: Hello World



- Create a new repository on github
  - Repository name: hello_world
  - Select public and add the README file
- Now to clone it in your home directory
  - On the terminal type ‘cd’ to make sure you are in your home
  - git clone https://github.com/<username>hello_world
  - Type ‘ls’ to confirm the new folder “hello_world” is there
- Congratulations on creating your first github repo!

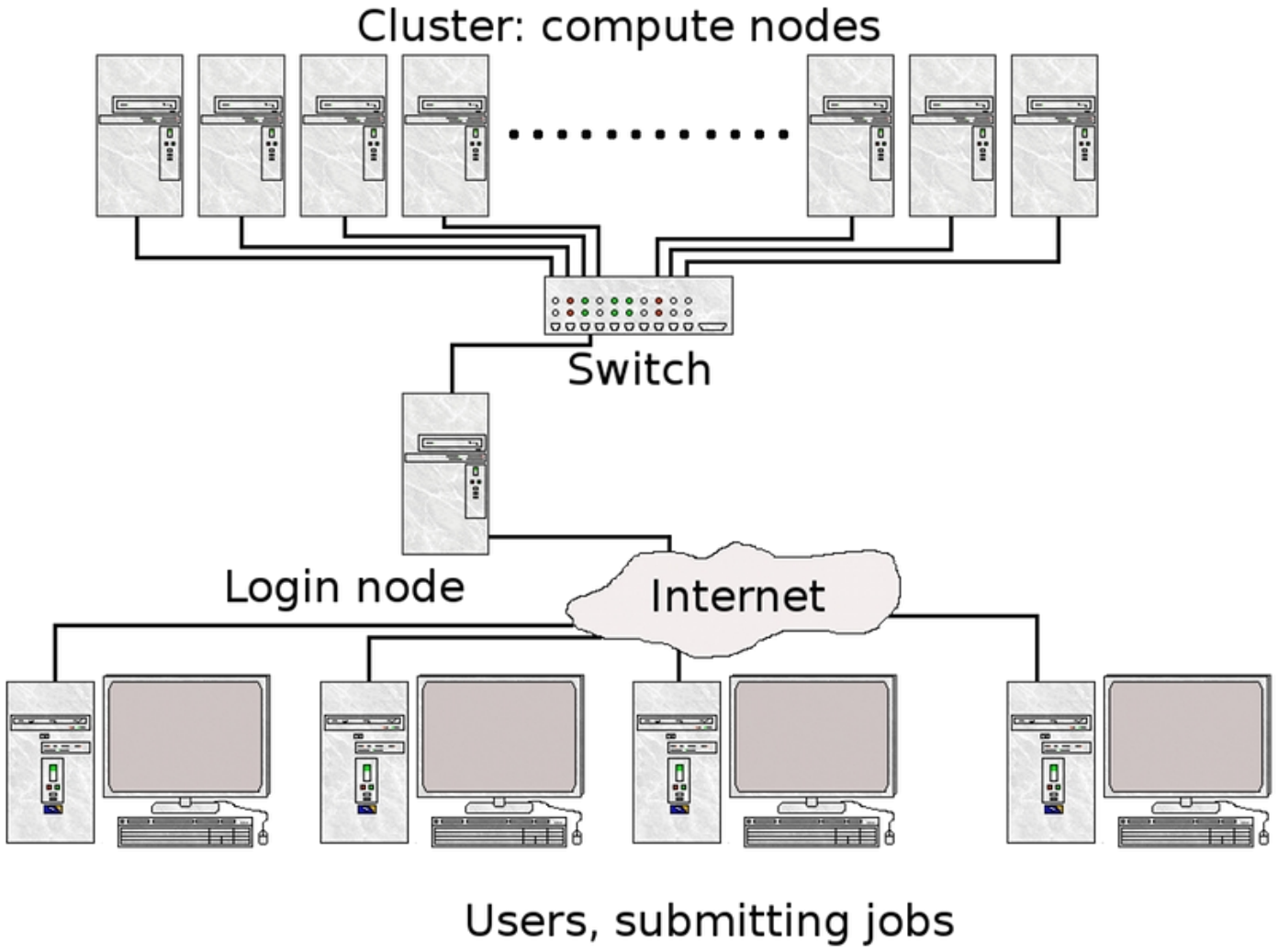
Github – Practice: Making Changes

- Change the README file:
  - cd hello_world
  - ls
- You should see README.md in the directory. With your favorite text editor change the readme file (vim, nano, gedit, echo…)
  - Add the line '## created by <your name>" and save the file
- Push your changes: (type 'git status' after each command)
  - git add README.md
  - git commit –m "Added author to README"
  - git push
- Now on your browser at https://github.com/<username>/hello_world you will see your updated README file

Clusters -

- Introduction to Cluster Computing

Clusters


Cluster: compute nodes
Switch
Login node
Internet
Users, submitting jobs

Why Clusters?

- Computers can generally do only one thing at a time. Multitasking is an illusion as the CPUs switch back and forth between tasks very fast
- Modern computers have multiple CPUs. Running a program with multiprocessing helps improve speed, but it has some limitations.
- A computer cluster is a set of computers that work together through a network. A cluster can be thought of as one giant supercomputer

Advantages to Clusters

1. High Performance :
The systems offer better and enhanced performance than that of mainframe computer networks.

2. Easy to manage :
Cluster Computing is manageable and easy to implement.

3. Scalable :
Resources can be added to the clusters accordingly.

4. Expandability :
Computer clusters can be expanded easily by adding additional computers to the network. Cluster computing is capable of combining several additional resources or the networks to the existing computer system.

5. Availability :
The other nodes will be active when one node gets failed and will function as a proxy for the failed node. This makes sure for enhanced availability.

6. Flexibility :
It can be upgraded to the superior specification or additional nodes can be added.

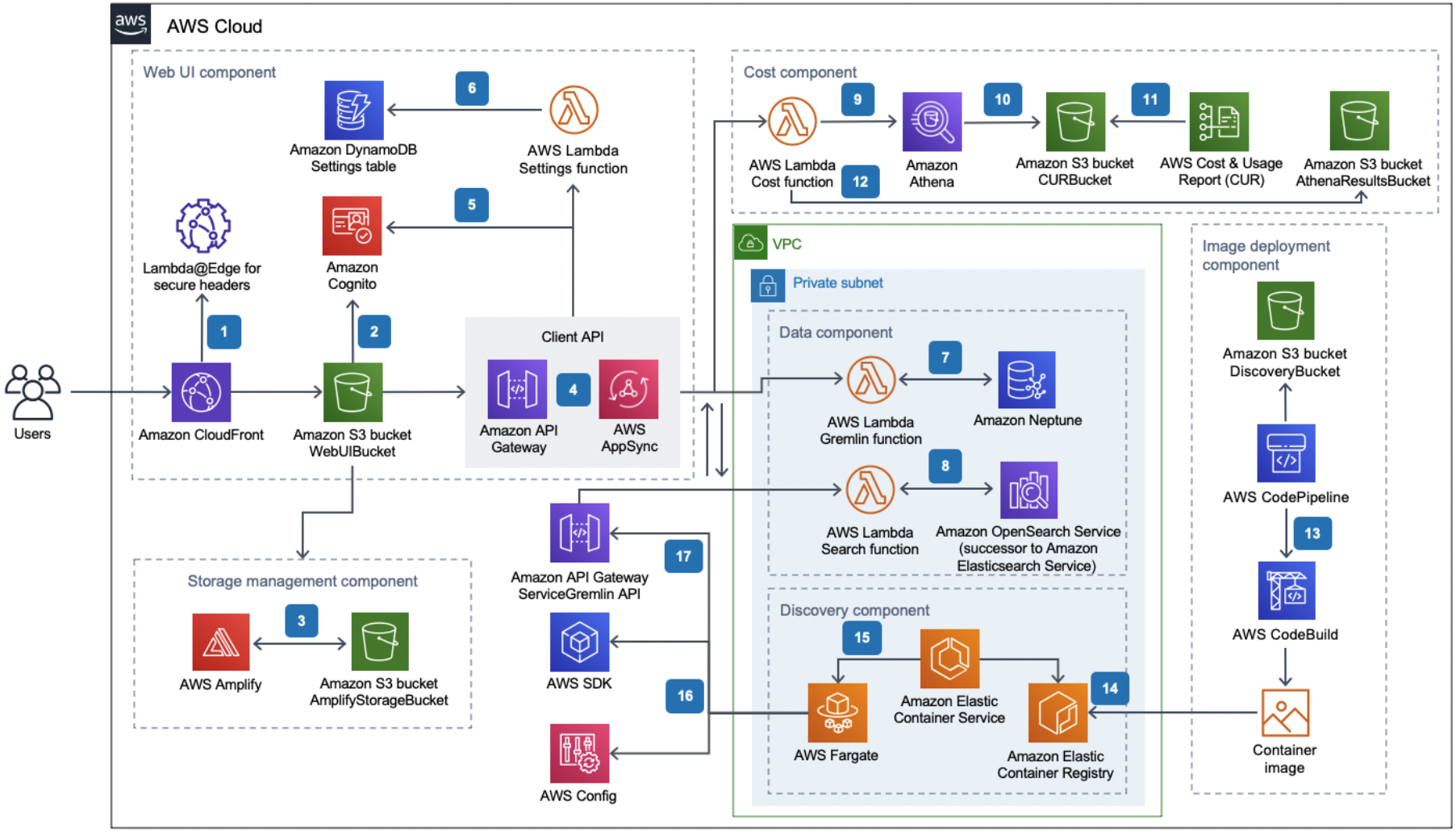
Disadvantages to Clusters

1.High cost :
It is not so much cost-effective due to its high hardware and its design.

2. Problem in finding fault :
It is difficult to find which component has a fault.

3. More space is needed :
Infrastructure may increase as more servers are needed to manage and monitor.

Cloud computing (AWS)

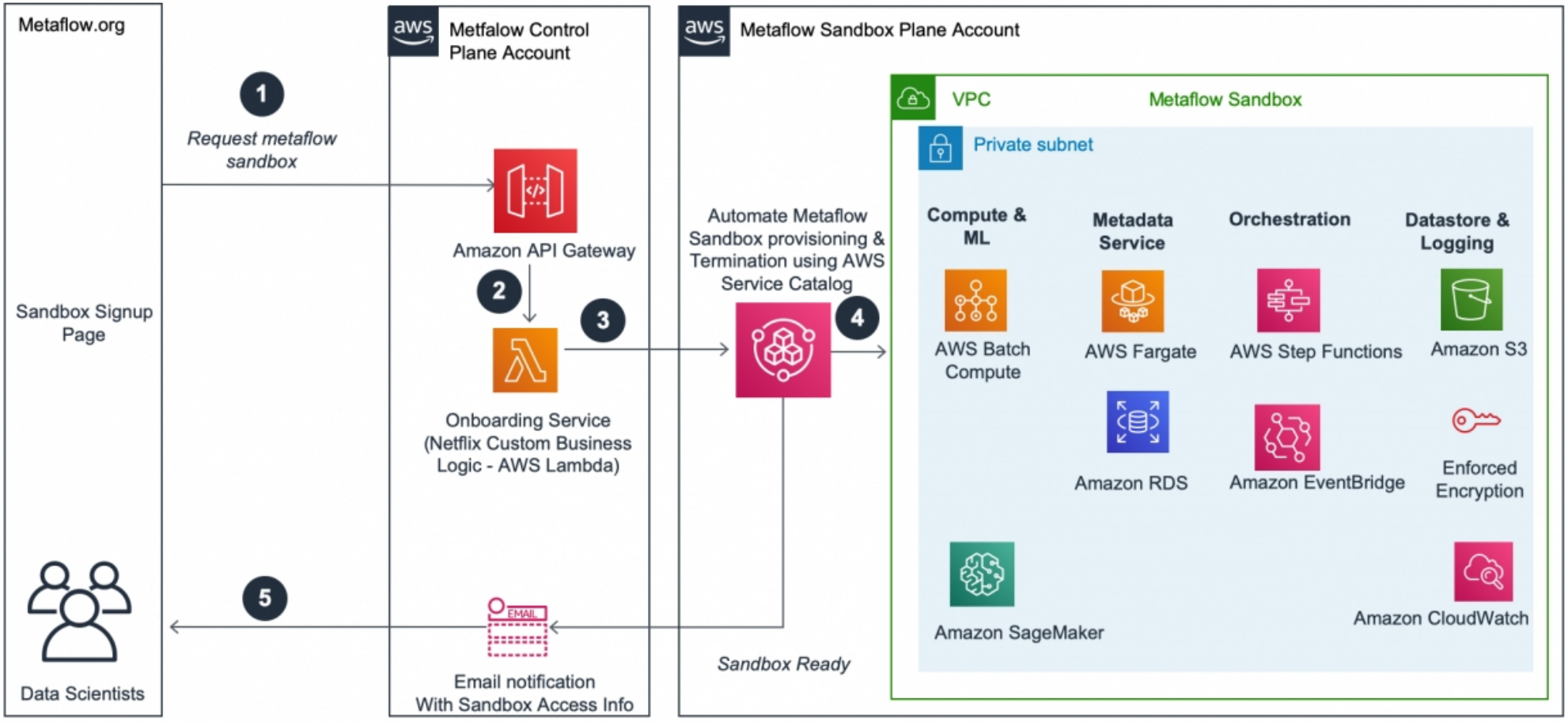


Cloud computing (AWS)

What is AWS?
AWS is a Cloud Computing platform, which helps you build your applications over the cloud. It offers various services like a combination of infrastructure and software services, along with computing power, scalability, reliability, and secure database storage. You can use AWS for quality development as it offers around 200 products and services all over the world.

The top 5 services provided by Amazon Web Services are:
Amazon Elastic Cloud Compute (EC2)
Amazon Simple Storage Service (S3)
Amazon Virtual Private Cloud (VPC)
Amazon CloudFront

Cloud computing (AWS - Netflix)



AWS - advantages

User-friendly
AWS is easy to use as the platform is specially designed for quick and secure access. Users can modify their data whenever they want, wherever they want.

Flexible
It always lets you use those operating systems, programming languages, and web application platforms that you are comfortable with.

Secure
Provides data protection, identity and access management, infrastructure protection, threat detection and continuous monitoring and compliance and data privacy.

Cost-effective
If you follow traditional methods, then you have build your own servers for storing your data and applications, which consumes a good amount of both your time and money.

Reliable
AWS serves over a million active clients in more than 200 nations all over the world

Highly Performant
High-performance computing (HPC) is the ability to process a massive amount of data at high speed.

Scalable and Elastic
If you want to add more servers, AWS enables you to use them within minutes.
If you use fewer resourcesthen AWS itself shrinks the resources to fit your requirement

AWS - disadvantages

Price Variations
The price of AWS services varies based on factors such as the cost of land, fiber, electricity, and taxes from region to region.

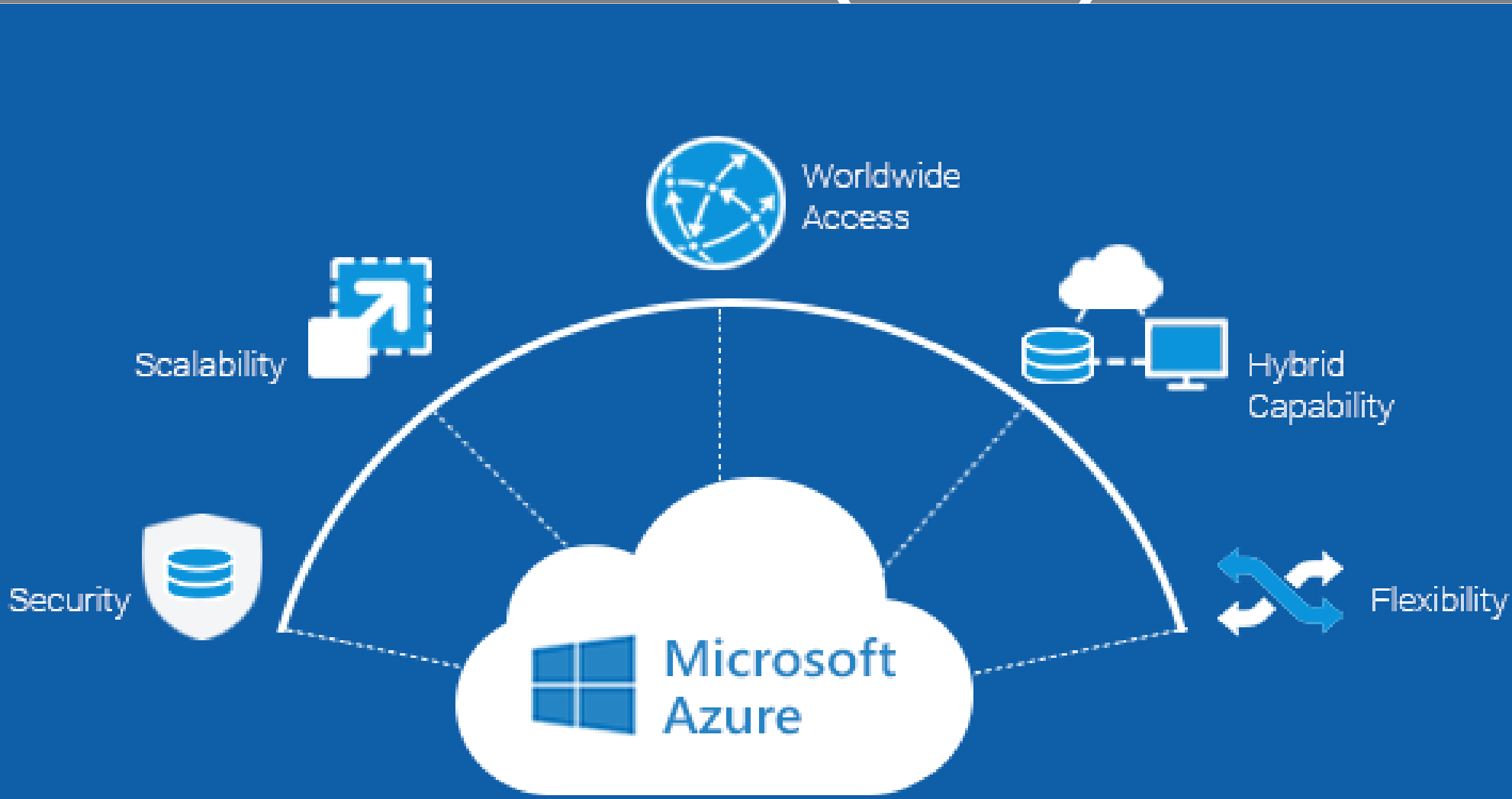
General Issues
Amazon is a huge family with millions of customers, so it has some temporary Cloud Computing issues.

Lack of Experts
However, only a few professionals are skilled in AWS or any cloud provider

Limitations
Companies that are using AWS will have default resources to use, but the problem comes when default resource limits vary from region to region.

Other Cloud services (Google)

Other Cloud services (Azure)



Other Cloud services (compare)

PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

Other Cloud services (compare)

Category	Service	amazon web services	Azure	Google Cloud Platform	IBM Cloud	ORACLE	Alibaba Cloud
Compute	**Shared Web hosting**		Azure shared App Services		Web hosting services		Web Hosting Simple Application Server
Compute	**Virtual Server**	Amazon EC2	Azure Virtual Machine	Compute Engine	Virtual Server Infrastructure (VSi)	Compute	Alibaba ECS
Compute	**Bare Metal Server**	Amazon EC2 Bare Metal Instance (Preview)	Azure Bare Metal Servers (Large Instance Only for SAP Hana)		Bare Metal Servers	Bare Metal Servers	ECS Bare Metal Instance
Compute	**Virtual Dedicated Host**	Amazon EC2 Dedicated Hosts		Sole Tenant Node (Beta)	Dedicated Virtual Servers Infrastructure (VSi)	Dedicated Compute Classic	Dedicated Host
Compute	**Container Registration Service**	Amazon EC2 Container Registry	Azure Container Registry	Container Registry	IBM Cloud Container Registry	Oracle Cloud Infrastructure Registry	Container Registry
Compute	**Container Management Service**	Amazon EC2 Container Service Amazon Elastic Container	Azure Kubernetes Service (AKS) Azure Container	Kubernetes Engine	IBM Cloud Kubernetes Service	Container Engine for Kubernetes (OKE)	Container Service Container Service for

Our Clusters

- UNCC has two main clusters:
  - Research Cluster
  - Education Cluster
- The Research clusters are for Faculty and Grad student researchers.
- The Educational clusters are a smaller version of the Research Clusters designed for education and the classroom environment.

The Education Clusters

- Centaurus is the main education cluster that we will use. It has 13 nodes.
  - 12 general compute nodes, each node has 16 dual Intel Xeon 3.2 GHz 8-core processors and 128 GB RAM
  - 1 large-memory node with a dual Intel Xeon 3.2 GHz 8-core processor and 768 GB of RAM
- The GPU cluster is also available with less nodes, but more GPUs for different kinds of processing. This one is usually faster when your application does a lot of math and stats processing.

Connecting to the Cluster

- We connect to the cluster through a terminal using an ssh (secure) connection. The clusters do not have a GUI (Graphical User Interface)
- All the clusters use Linux. This is generally true for most clusters in any environment. There are some exceptions, but most are at least Unix based. Only a madman would run a cluster with Windows computers.
- For security, there are a couple of requirements to connect to the clusters at UNCC.
  - Have Duo enabled for your account. As of March 16, 2021 all students are supposed to have Duo. If for some reason you don't have Duo, here is a link with a tutorial to sign up: https://spaces.uncc.edu/pages/viewpage.action?pageId=35651686
  - Be either connected directly to UNCCs network (eduroam or directly with an Ethernet cable) or connect to UNCCs VPN.
  - https://spaces.uncc.edu/pages/viewpage.action?pageId=6653379

Connecting to the Cluster

- After connecting to VPN or directly on UNCCs network, open a terminal and type:
  - ssh <username>@hpc-student.uncc.edu
- You will be prompted to enter your password. Nothing will show up as you type, not even *****, but be assured that it is accepting your input.
- You will then get a prompt asking how you would like to authenticate with Duo. Selecting the first option (press ‘1’ and enter) will send a push notification to your phone

Running Your First Job

- Once you are authenticated on the cluster you can beginning executing commands and submitting jobs.
- You should not execute long running jobs directly from the node that you first log into. This is more of a “Welcome Node” where you can configure your environment, write code, upload files, and small tests.
- **Slurm** is a scheduler for big jobs. Basic steps to submit a job (example on next slide):
  - Create a **slurm** bash script (contains options for your script line job-name, # of nodes...)
  - Create your executable script, or execute other programs directly from the slurm script.
  - In the terminal, run the command:
    - sbatch slurm.sh
  - The command “sacct” or “squeue -u <user>” can be used to check the status of your job.

Running Your First Job

cluster_script.sh

```
#!/bin/bash

sleep 1
echo "Job# $1 from $(hostname)"
```

cluster-slurm_sbatch

```
#!/bin/bash

#SBATCH --partition=Centaurus          # Partition name (Centaurus or GPU)
#SBATCH --job-name=script_test         # Job name
#SBATCH --nodes=5                      # Number of total nodes (computers on the cluster)
#SBATCH --mem=1gb                      # Memory per node
#SBATCH --time=0:10:00                 # Time limit hrs:min:sec OR days-hrs
#SBATCH --output=out-%x-%j.log         # Standard output and error log


SECONDS=0

for i in $(seq 5)
do
    echo "Starting job: $i"
    srun --nodes=1 --exclusive ./script.sh $i &
done
wait

echo "Runtime is: $SECONDS seconds"
```

Quiz 13

- On canvas now