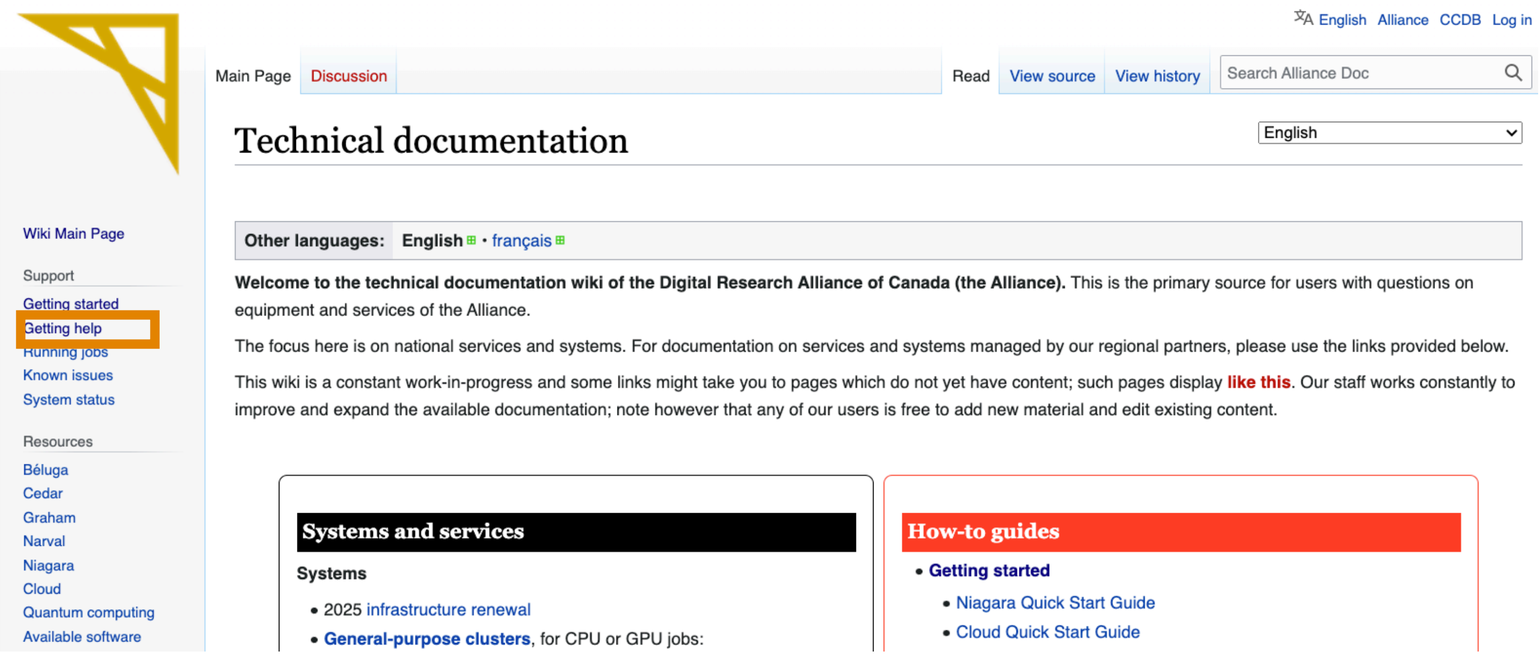
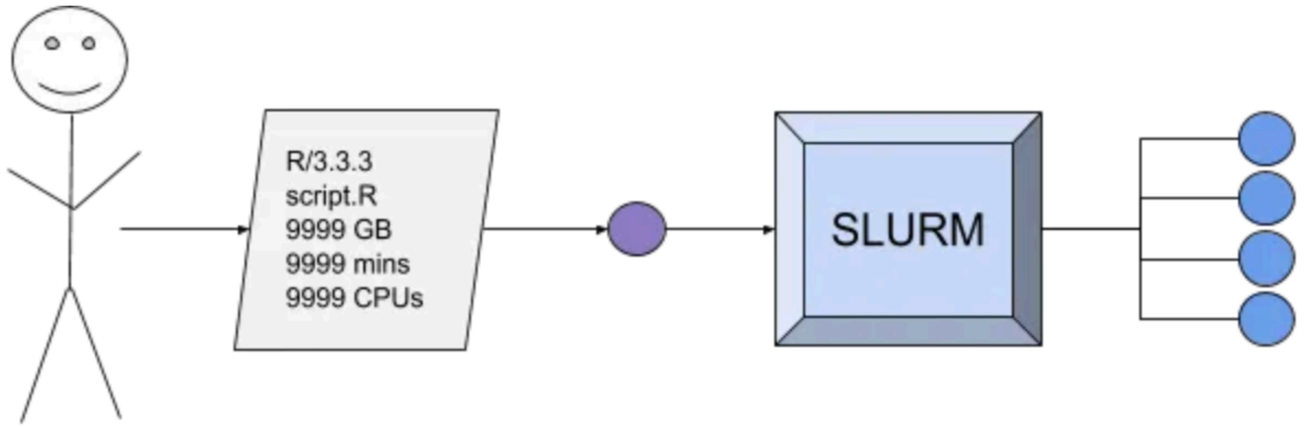
Alliance Canada wiki & Tech Support

- Alliance Canada wiki
  - FAQ
  - Step-by-step instructions
  - Workshops that go deeper than this one
- Tech Support ("Getting help")
  - Read their instruction before sending email
  - Respond pretty fast if email is proper



The general workflow

- Apply for an account if you haven't!
- Upload your script to one of these general-purpose clusters: Béluga, Cedar, Graham, Narval
  - Make sure your code runs! Debugging is not straightforward here
  - (Find the bottleneck and improve the efficiency of your code)
  - Make sure all input/output streams use relative paths
- Create connection with login node, the "receptionist"
- Before the first time, prepare the materials to run your code
  - For example `>install.packages( ‘‘Rcpp’’ )`
- Submit job
  - You specify the time, number of CPUs, memory, the command you want to run
  - The "scheduler" will decide when to do it and allocate the resource
  - Nodes "workers" will receive and complete your "order"
- Wait...
- Don't forget to monitor and check if the job is done
- Download the output



user	job	login node	scheduler	nodes
That's you!	A small file you write that tells the computer what program and script to run and describes the computational power needed for the job. - Memory - Duration - Number of cores - Serial or parallel	A computer that lets you connect to the rest of the Compute Canada computers. You can look at your files on this node, and use it to submit jobs to the scheduler.	An algorithm (called SLURM) that decides when your script will run, based on how long it will take/how many resources it needs.	The scheduler sends your job to one (or many) nodes. These nodes are the computers that will be running your script. Each node has CPUs, RAM and storage.

https://medium.com/the-nature-of-food/how-to-run-your-r-script-with-compute-canada-c325c0ab2973

Use keyboard & Travel along the paths

- Some shortcuts:
  - arrow keys [up/down]
  - `ctrl+A` [go to the start of line]
  - `ctrl+E` [go to the end of line]
  - `ctrl+C` [stop the current process]
- Useful tools to save some typing
  - wildcard: `*` [Match any character]
  - `tab` [automatically complete path]
  - `tab*2` [show all options for ambiguous path]
- Some commands:
  - `pwd` [print working directory]
  - `ls` [listing items]
  - `cd` [change to directory]
- Paths
  - Absolute paths begin with `/`
  - Relative paths start from current directory
  - `.` [Current directory]
  - `..` [Parent directory]
  - `~` [Home directory]

Terminal

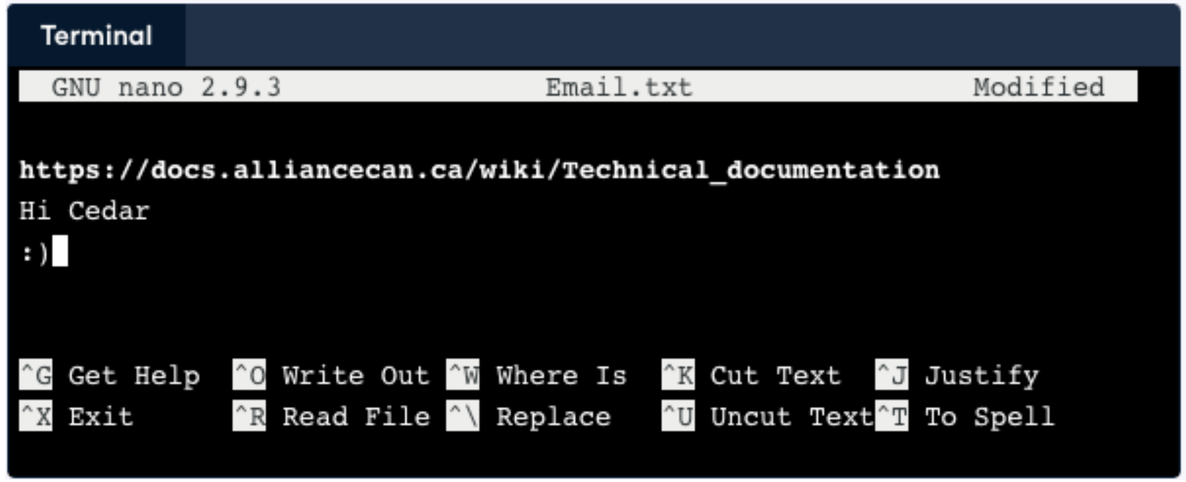
```
$ cd ~
$ pwd
/home/repl
$ ls
backup  bin  course.txt  people  seasonal
$ ls /home/repl/people
agarwal.txt
$ cd ./seasonal
$ ls s*.csv
spring.csv  summer.csv
$ cd ../people
$ pwd
/home/repl/people
$
```

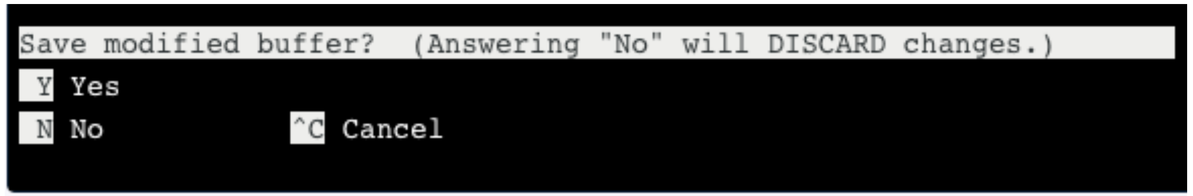
Practice

1. Download the sandbox folder and record its location
2. Open your command lines software
3. Change working directory to the sandbox folder (try the shortcuts to navigate in line). If your folder name contains spaces you should use quotes for the path. What is different before the dollar sign?
4. List all items in sandbox
5. Change directory to “people” and print working directory.
6. Change directory to the folder in its parent folder beginning with “se” (use tab completion)
7. List all csv files that end with “r.csv”
8. Change working directory back to Sandbox without using the absolute path

Modify the files

- Modify files and directories
  - `cp` [copy]
    - `cp course.txt course_new.txt`
    - `cp spring.csv autumn.csv people`
  - `mv` [move or rename]
    - `mv spring.txt autumn.txt ..`
    - `mv course.txt course-A.txt`
  - `rm` [remove] ⚠️
    - `rm course_new.txt`
  - `rmdir` [remove directory; only delete empty]
  - `mkdir` [?]
    - `mkdir year`
- Create a file
  - `touch Email.txt` [create a file without editing]
  - `nano Email.txt` [text file edit; create if not exist]





Practice

1. Type “`clear`” to clear terminal window
2. In sandbox, create a folder called “backup” with a command
3. Copy the text file in “people” to “backup”.
4. Rename the new copy in “backup” as “people_backup.txt”
5. Use the text editor to create “wiki.txt” in “people”. It contains the address to Alliance Canada wiki.
6. Move “wiki.txt” to backup, overwriting “backup/people_backup.txt”

Print something out & More handy tricks

- Print something out
  - `cat` [look at the content of some file]
  - `echo` [print]
  - `head` [look at the beginning of some file]
  - `tail` [?]
- General structure: <Command> <-options> <arguments>
  - `head -n 3 course.txt`
- man [manual of commands] [:q to quit]
  - `man head`
- `>` [store output in a new file]
  - `head -n 3 course.txt > Files.txt`
- `>>` [append output to a file]
  - `ls seasonal >> Files.txt`
- `|` [pipe: pass the output as the input of the next command]

Practice

1. Print “Hello HPC” in the terminal.
2. shell has environment variables, USER is one of them. To get its value it has to follow a $. Try `echo $USER` and `echo USER`
3. Look at the head, tail and content of course.txt
4. `head -n 5 course.txt|tail -c 8` Look at the manual to figure out what does it do. Which word will it print? Run it to check.
5. Collect the last 3 lines of all files in “seasonal” and save in “seasonal/Tails.txt”
6. Challenge: look at the content of “seasonal/Tails.txt”. The repetitive headers are very annoying! Can you find out which flag of tail can remove it ?

Introduction to Bash scripting

- Nano script.sh and write some commands in it.
- If you submit the sh file, then the shell just run the commands
  - “bin” the location of executable files.
  - “bash” Bourne Again Shell, is an app

EXP...

script.sh

server_log_wit...

script.sh ×

```
1  #!/bin/bash
2
3  # Concatenate the file
4  cat server_log_with_todays_date.txt
5
```

```
repl:~/workspace$ cat server_log_with_todays_date.txt
2019-01-01 | server request | windows | ping-2.000
2019-02-01 | server request | mac | ping-2.000
2019-02-01 | server request | windows | ping-12.000
2019-02-01 | server request | linux | ping-6.000
repl:~/workspace$ bash script.sh
2019-01-01 | server request | windows | ping-2.000
2019-02-01 | server request | mac | ping-2.000
2019-02-01 | server request | windows | ping-12.000
2019-02-01 | server request | linux | ping-6.000
```

How to connect to AllianceCan clusters

- ssh [secure shell, a standard to connect to remote machines securely]
  - `ssh user@cluster.alliancecan.ca`
  - you will be asked to input your password and multifactor authentication

```
Hilda:Sandbox hildalyn$ ssh xz424@cedar.alliancecan.ca
(xz424@cedar.alliancecan.ca) Password:
(xz424@cedar.alliancecan.ca) Duo two-factor login for xz424

Enter a passcode or select one of the following options:

 1. Duo Push to iPhone (iOS)

Passcode or option (1-1): 637448
Success. Logging you in...
Success. Logging you in...
-------------------------------------------------------------------------------
===============================================================================
Welcome to Cedar! / Bienvenue sur Cedar!

For information see: https://docs.alliancecan.ca/wiki/Cedar
Email support@tech.alliancecan.ca for assistance and/or to report problems.
```

```
[xz424@cedar1 ~]$ _
```

Submit a job

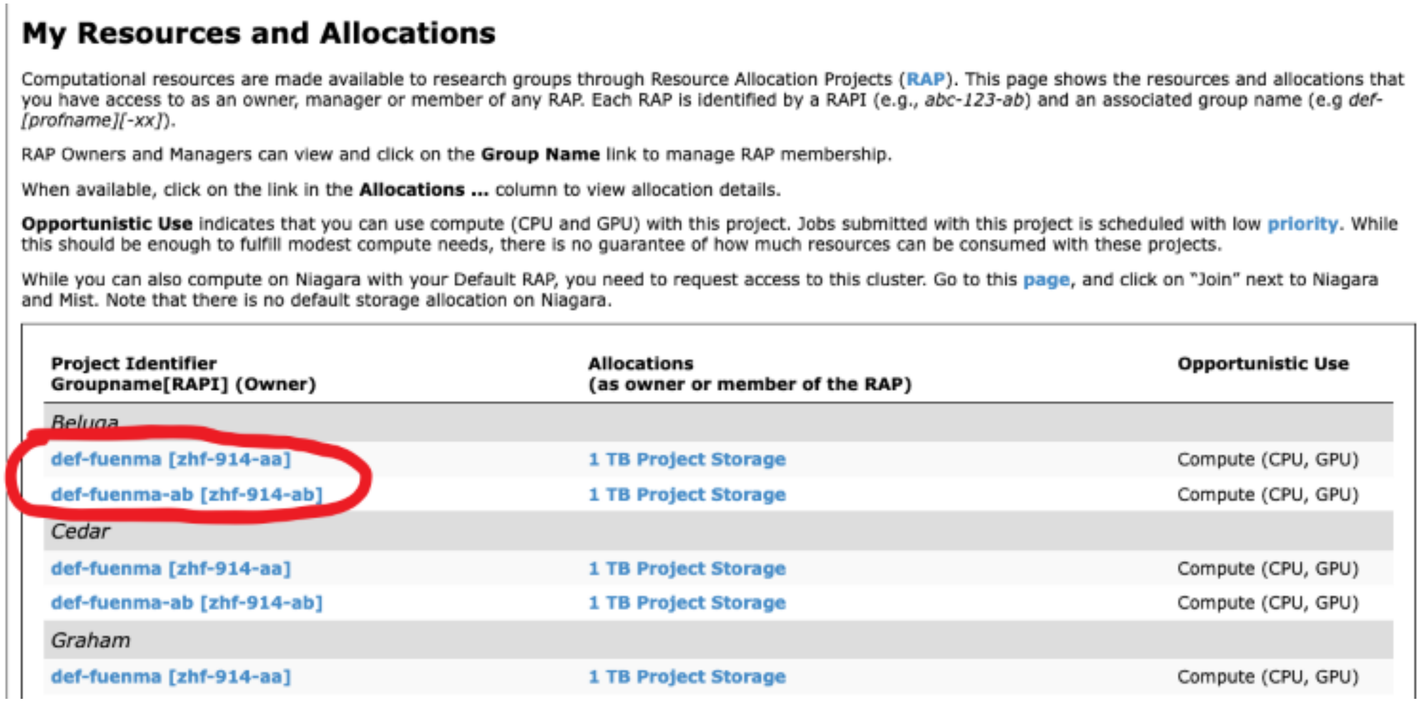
- `sbatch` [submit a job]

```
$ sbatch simple_job.sh
Submitted batch job 123456
```

  - simple_job.sh [a minimal example]

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --account=def-someuse
echo 'Hello, world!'
sleep 30
```

  - On general-purpose clusters, this job reserves 1 core and 256MB of memory for 15 minutes.
  - account: log in to CCDB; My Account -> My Resources and Allocations.
  - you will receive a jobID such as 123456



My Resources and Allocations

Computational resources are made available to research groups through Resource Allocation Projects (RAP). This page shows the resources and allocations that you have access to as an owner, manager or member of any RAP. Each RAP is identified by a RAPI (e.g., *abc-123-ab*) and an associated group name (e.g *def-[profname][-xx]*).

RAP Owners and Managers can view and click on the **Group Name** link to manage RAP membership.

When available, click on the link in the **Allocations ...** column to view allocation details.

Opportunistic Use indicates that you can use compute (CPU and GPU) with this project. Jobs submitted with this project is scheduled with low priority. While this should be enough to fulfill modest compute needs, there is no guarantee of how much resources can be consumed with these projects.

While you can also compute on Niagara with your Default RAP, you need to request access to this cluster. Go to this page, and click on "Join" next to Niagara and Mist. Note that there is no default storage allocation on Niagara.

Project Identifier Groupname[RAPI] (Owner)	Allocations (as owner or member of the RAP)	Opportunistic Use
Beluga		
def-fuenma [zhf-914-aa]	1 TB Project Storage	Compute (CPU, GPU)
def-fuenma-ab [zhf-914-ab]	1 TB Project Storage	Compute (CPU, GPU)
Cedar		
def-fuenma [zhf-914-aa]	1 TB Project Storage	Compute (CPU, GPU)
def-fuenma-ab [zhf-914-ab]	1 TB Project Storage	Compute (CPU, GPU)
Graham		
def-fuenma [zhf-914-aa]	1 TB Project Storage	Compute (CPU, GPU)

```
#!/bin/bash
#SBATCH --account=def-someuser   # replace this with your own account
#SBATCH -mem-per-cpu=4G        # memory per cpu
#SBATCH -time=0-01:00           # time (DD-HH:MM)
#SBATCH -cpus-per-task=4
module load gcc/9.3.0 StdEnv/2023 r/4.3.1   # modules



Rscript Slow.R > stdout.txt 2>stderr.txt
```

Monitor current jobs

- `squeue` [to run or running]
  - `squeue -u $USER` [all my jobs the scheduler is managing, including running and pending]
  - `squeue -u $USER -t RUNNING` [?]

```
$ sq
   JOBID     USER      ACCOUNT      NAME  ST   TIME_LEFT NODES CPUS    GRES MIN_MEM NODELIST (REASON)
  123456   smithj   def-smithj  simple_j   R        0:03     1    1  (null)      4G cdr234  (None)
  123457   smithj   def-smithj  bigger_j  PD  2-00:00:00     1   16  (null)     16G (Priority)
```

Cancel a job

- `scancel 1234567` [cancel jobID 1234567]
- `scancel -u $USER` [cancel all my jobs running or pending]
- `scancel -t PENDING -u $USER` [cancel my pending jobs]

Summarize completed jobs

```
$ seff 12345678
Job ID: 12345678
Cluster: cedar
User/Group: jsmith/jsmith
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 02:48:58
CPU Efficiency: 99.72% of 02:49:26 core-walltime
Job Wall-clock time: 02:49:26
Memory Utilized: 213.85 MB
Memory Efficiency: 0.17% of 125.00 GB
```

Practice

If you have set up the account, submit simple_job.sh, monitor its status. You can cancel it, or check it when it's finished.

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --account=def-someuser
echo 'Hello, world!'
sleep 30
```

- https://docs.alliancecan.ca/wiki/Getting_started
- https://docs.alliancecan.ca/wiki/Technical_support
- https://docs.alliancecan.ca/wiki/Utiliser_des_modules/en
- https://docs.alliancecan.ca/wiki/Running_jobs
- Datacamp