Jun 13, 2025

Research Cluster: Spineless Genomics

DOI
dx.doi.org/10.17504/protocols.io.eq2lypw9plx9/v1

Marina McCowin[1]

[1]ucsd

Rouse Lab

Marina McCowin
UCSD- Scripps Institution of Oceanography

DOI: **dx.doi.org/10.17504/protocols.io.eq2lypw9plx9/v1**

Protocol Citation: Marina McCowin 2025. Research Cluster: Spineless Genomics. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.eq2lypw9plx9/v1

Protocol status: Working
We use this protocol and it's working

Created: December 21, 2020

Last Modified: June 13, 2025

Protocol Integer ID: 45795

Abstract

Brief explanation for how to access, log in, and navigate the research cluster server. Note that this protocol is optimized for Mac users. Windows users will likely need to adjust to running commands in Powershell. See cluster user guide for help: https://collab.ucsd.edu/display/RESUP/UCSD+Research+Cluster%3A+User+Guide

More information about the cluster can be found here: **https://blink.ucsd.edu/faculty/instruction/tech-guide/dsmlp/**

Research cluster contact: datahub@ucsd.edu
We also have a contact for more specific/urgent technical help, Alan Moxley (reach out to Marina for contact info).

1. BEFORE USING

1 Log on to the UCSD VPN with your AD login credentials (Cisco AnyConnect if you're not on UCSD campus Wifi). You will need to be connected to the VPN (or be on UCSD Wifi) in order to access the cluster.

2 If you are interested in additional information about the cluster, or get stuck and need help, check out the User Guide, available here:
https://collab.ucsd.edu/display/RESUP/UCSD+Research+Cluster%3A+User+Guide

You can also check the overall status of the cluster here:
https://datahub.ucsd.edu/hub/status
Sometimes it gets very busy at the end of the quarter, so this is a good place to check if you're having issues launching pods due to a lack of available resources.

3 Open a terminal window. There are *two* ways to log on to the server, one for uploading/downloading files, and one for running pods and programs. If you like using Jupyter Notebooks, that is also an option (link available when you launch a pod).
Navigate to the following sections according to your needs:
- Section 2: Uploading & Downloading Files (sftp)
- Section 3: Running Pods and Programs (ssh)
- Section 4: Accessing and Deleting Pods (ssh)
- Section 5: Checking Progress of Processes (ssh)
- Section 6: Using MitoFinder on the Cluster (ssh)

2. Uploading and Downloading Files (sftp)

4 Your login credentials will be very similar to your AD login credentials/UCSD email. To Log in to the server for the purposes of uploading or downloading files, use the **sftp** command and your AD login appended by **@dsmlp-login.ucsd.edu**, then press enter.

Command

```
sftp adlogin@dsmlp-login.ucsd.edu
```

You will be prompted for a password. Type in your AD password and press enter.

5 Once you're logged into the server, your automatic working directory will be your home directory (on the server). Use **ls** to view folders and files in your home directory and **cd** to navigate in and out of folders as you would on your home computer.

6 To upload a file from your computer to the server, use **put** with the absolute path of your file on your computer (Mac users can click and drag their file into the terminal window to copy the file's absolute path).

You can also use the point-and-click interface available through Jupyter notebooks to upload files (look for a link when you launch a pod; see Section 3).

Command

```
put absolutepath/to/file/filename
```

If your files are large, they may take some time to upload. Once finished, use **ls** to double check that your file was uploaded.

7 To download a file from the server, use **get** with the path of the file you want *on the server*. If you don't specify a download location (where you want it to go), it will appear in your home directory on your computer (or whatever your default working directory is). You can specify a location on your computer where you want the file to go by adding the absolute path to that folder on your computer *after* the path to the file you want on the server, as you see below.

Command

```
get path/to/file/filename path/to/local/folder
```

If you'd like to download an entire folder rather than just a single file, add **-r** to the get command.

Command

```
get -r path/to/folder
```

8 When you're done uploading files, use the **exit** command to exit the server.

Command

```
exit
```

3. Running Pods and Programs (ssh)

9 Your login credentials will be very similar to your AD login credentials/UCSD email. To Log in to the server, use the **ssh** command and your AD login appended by **@dsmlp-login.ucsd.edu**, then press enter.

Command

```
ssh adlogin@dsmlp-login.ucsd.edu
```

10 In order to install and run programs from the command line, you'll need to launch a pod (this allocates cluster resources like cores and memory, to you). The basic command to launch a pod is:

Command

```
launch-scipy-ml.sh
```

Launching a pod with this command (as is, no arguments) will provide a default allocation of 2 cores and 8 GB of RAM. You can request up to 8 cores and 64GB of RAM for a single pod using the options **-c** (specify number of cores) and **-m** (specify GB of RAM). For example, you could launch a pod with maximum resources with the following:

Command

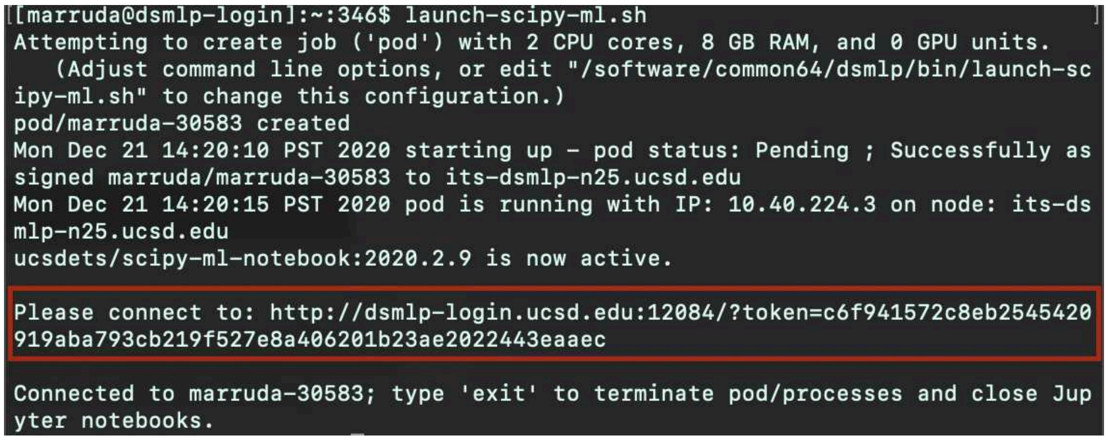
```
launch-scipy-ml.sh -m 64 -c 8
```

Other detailed options for launching pods are available (see research cluster User Guide: https://collab.ucsd.edu/display/RESUP/UCSD+Research+Cluster%3A+User+Guide#UCSDResearchCluster:UserGuide-LaunchScriptCommandLineOptions)

11 ***JUPYTER NOTEBOOKS***

Once you've launched a pod, you will be provided with a Jupyter Notebooks link. If you're interested in using Jupyter Notebooks, paste the link into your browser. Otherwise all analyses/processes can be run from the command line in your terminal window. *Note that you* cannot *access Jupyter Notebooks upon pod-launch if you're running all your pods "in the background" (more on background pods in Step 13).*

Here's an example of what this would look like (link in the box will change every time you launch a pod):



```
[marruda@dsmlp-login]:~:346$ launch-scipy-ml.sh
Attempting to create job ('pod') with 2 CPU cores, 8 GB RAM, and 0 GPU units.
   (Adjust command line options, or edit "/software/common64/dsmlp/bin/launch-sc
ipy-ml.sh" to change this configuration.)
pod/marruda-30583 created
Mon Dec 21 14:20:10 PST 2020 starting up - pod status: Pending ; Successfully as
signed marruda/marruda-30583 to its-dsmlp-n25.ucsd.edu
Mon Dec 21 14:20:15 PST 2020 pod is running with IP: 10.40.224.3 on node: its-ds
mlp-n25.ucsd.edu
ucsdets/scipy-ml-notebook:2020.2.9 is now active.

Please connect to: http://dsmlp-login.ucsd.edu:12084/?token=c6f941572c8eb2545420
919aba793cb219f527e8a406201b23ae2022443eaaec

Connected to marruda-30583; type 'exit' to terminate pod/processes and close Jup
yter notebooks.
```

Red box shows Jupyter Notebooks Link. Paste this into your browser to use the Jupyter Notebooks interface.

12 Now that you've launched a pod, you can run whatever lines of code you need to install programs and their dependencies, run programs, etc (all from the command line)! If you're interested in using MitoFinder, that program and all of its dependencies are already pre-installed on the cluster for you. You should see them in your home directory (on the server) if you use the command **ls** to view the contents of your directory after launching a pod.

13 ***IMPORTANT NOTE:*** If you do not maintain your connection with the server (e.g. your VPN connection times out, wifi dies, computer goes to sleep, etc), it may interrupt your analysis!

In order to avoid this, you can run launch your pod in the "background" with the **-b** argument. This argument will still allocate the resources you request when you launch your pod, but will run everything in the background, so the system no longer requires a direct connection to your computer (you can disconnect from wifi, shut down your computer, etc and still return later to check on your analysis).

Command

```
launch-scipy-ml.sh -m 63 -c 7 -b
```

The above script would launch a pod with 63GB of RAM, 7 cores, and would run *in the background*, allowing you to interrupt your connection with the server without interrupting any processes you have running on it.

More information about running pods in the background here:
https://collab.ucsd.edu/display/RESUP/UCSD+Research+Cluster%3A+User+Guide#UCSDResearchCluster:UserGuide-BackgroundExecution/Long-RunningJobs

4. Accessing and Deleting Pods (ssh)

14 To check how many pods you currently have running on the server, use the command **kubectl get pods** after logging in. This is also a great way to get the pod names (for pods launched in the background) so that you can access them.

Command

```
kubectl get pods
```

Note that you can launch as many pods as you like, provided you do not exceed 64GB of total RAM and 8 total cores in your combined total resource allocations for them (each student in our group has 64GB RAM and 8 cores at their disposal).

15 To access a different pod or one you have running in the background, you'll need the pod name (see above) and the **kubesh** command. Once you've "entered" the pod with **kubesh**, you can run processes/programs from the command line as normal.

Command

```
kubesh adlogin-12345
```

Pod names will consist of your AD login and a 4-6 digit number (12345 is just an example), separated by a hyphen.

16 To stop running a pod and return the resources allocated to it to the cluster, use **kubectl delete pod** and the appropriate pod name.
Please note: ***This does NOT delete ANY of your data!*** It simply ends any *processes* you have running and returns the memory and cores allocated to the cluster (analogous to

quitting a program after saving; any files created already will have automatically saved on the cluster).

Command

```
kubectl delete pod adlogin-12345
```

Note

Note: If you are only running a single pod (no pods in the background) you can also delete it (terminate the processes) using the **exit** *command (you do not need to specify the pod name).*

Please remember to ***ALWAYS DELETE PODS WHEN YOU'VE FINISHED*** running your processes so that those resources can be allocated to other users on the cluster.

See the User Guide for detailed information about time limits:
https://collab.ucsd.edu/display/RESUP/UCSD+Research+Cluster%3A+User+Guide#UCSDResearchCluster:UserGuide-ContainerRunTimeLimits

5. Checking Progress of Processes

17

Note

There are multiple ways to check the progress of processes you're running: you can check what pods are active, you can check what processes are currently active, and you can view the log files of processes you're running.

Regardless of what you'd like to check, you'll need to make sure you're logged into the server:
If you're logged out of the server, log back in with ssh (see step 9).

18 **To check active pods:**

(You may want to do this as a quick check to make sure your pods are still active if you logged out/lost connection or it's been a while since you started a background pod)

Once you're logged in to the server (ssh), use the **get pods** command to view active pods.

Command

```
kubectl get pods
```

This will print a list of active pods to your terminal. Active pods should say "Running" next to the pod name.

19 **To check active processes:**
(You may want to do this to check what stage an analysis is in, i.e. which programs are running at the present moment)

You will need to move into the active pod (if applicable) in order to view processes active within it. To do this, use the **kubesh** command with the pod name (use the **kubectl get pods** command from above if you can't remember the name of the pod you would like to check).

Command

```
kubesh adlogin-12345
```

Once you're in your active pod, use the **htop** command to view current activity in that pod (similar to the Activity Monitor on a Mac).

Command

view activity in local environment

```
htop
```

If your Terminal window is small, you'll need to expand it (at least half your screen) to easily see the programs that are running. A list of actively running programs can be found at the bottom of the window, past all of the bars. In the "Command" column you'll see green text that lists the name of each command running (note that this includes the absolute path to the file, so if you're running MitoFinder, look for commands with paths into the MitoFinder folder).

You can quit the activity monitor window by pressing **q** or F10.

20 **To check log files**:
(This is a quick way to view files in your terminal window without downloading them. You can edit files in this way too, so if you're just *interested in checking the progress of a log file, be careful not to make any edits)*

Once you're logged in (ssh), use the **ls** command to view the files in your current working directory.

Command

list files and directories

```
ls
```

Running the above command will print a list of all the files/directories in your working directory on the cluster. To view one, you'll use the command **view**. If you're looking for a mitofinder log, you'll be looking for a file that is some variant of "MitoFinder.log"

Command

view and/or edit a file

```
view name_of_file.log
```

Running the above command will make the file contents visible in your terminal window (scroll up and down to view all contents, remember to be careful not to make any edits if you are viewing an in-progress file).

When you're ready to leave the view window and return to your normal terminal window, run the **:q!** command to quit the viewer (this will not quit the terminal).

Command

quit the file viewer/editor

```
:q!
```

6. MitoFinder

21 MitoFinder can be pre-loaded onto the cluster using Docker (image including MitoFinder and all of its dependencies already created). To access the Docker image, add the following code to your launch script when you launch your pod:

Command

Add to launch script to use MitoFinder

```
-i amoxley/mitofinder:2020.10.28-3
```

If you choose to run your pod in the background (as in **Step 13**), your launch script will look something like this:

Command

Launch a pod (with MitoFinder) that will use no more than 63GB of memory, 7 cores, and will run in the background

```
launch-scipy-ml.sh -i amoxley/mitofinder:2020.10.28-3 -m 63 -c 7 -b
```

22 ***NOTE:*** *If you've launched your pod in the background (as in the example above) you will need to navigate into it in order to run MitoFinder.*

To do this, use the command **kubesh** and the pod name as in **Step 15** (pod name will be auto-generated and printed to your terminal once the pod launches successfully). For example, your code will look something like this (pod name will be specific to you):

Command

```
kubesh adlogin-12345
```

23 Once you've navigated into your pod, you can run MitoFinder as normal. The absolute path to the program is **/opt/MitoFinder/mitofinder** so you can use that to call the program as you normally would.

Consider the example below, but *remember to customize the MitoFinder settings to your specifications* for your project (e.g. metaspades vs megahit, memory limits, etc). Do not simply run this code without first understanding which settings you need for your assembly. The output folder with results will appear in your home directory on the cluster.

Command

Example code for running MitoFinder. Remember to provide an output folder name (samplename_1 in this example), and the fq.gz-format files for your trimmed reads, as well as a .gb-format reference genomes file. This example is set to run with a maximum of 63GB of RAM and 7 cores.

```
/opt/MitoFinder/mitofinder --megahit -t mitfi -j samplename_1 -1
sampletrimmed_P1.fq.gz -2 sampletrimmed_P2.fq.gz -r
reference_mitogenomes_file.gb -o 5 -m 63 -p 7
```