Zero to peaks

This tutorial will walk through setup, installation and running the eCLIP pipeline. For this tutorial, we will be spinning up an AWS ec2 instance with recommended hardware settings.

1. Setting up AWS

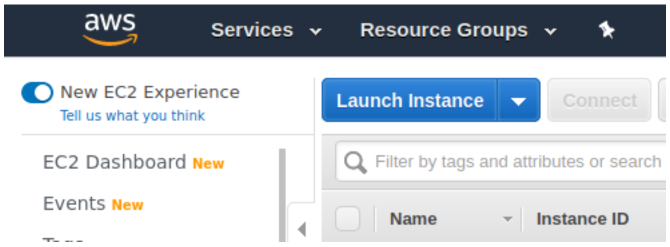
~5 - 20 minutes depending on your level of experience with Amazon web services

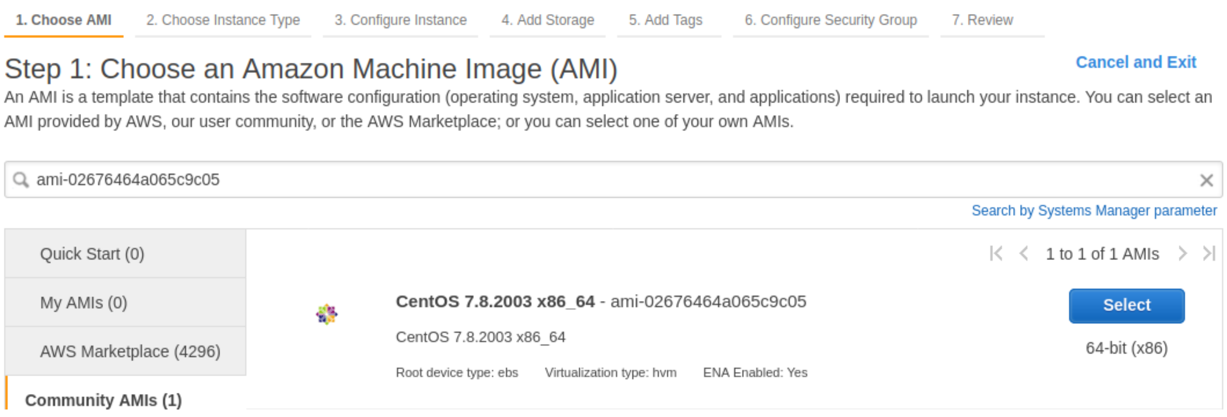
Launch an EC2 instance

The easiest way to launch an instance is through the console. Login and click "Launch Instance" to start the setup wizard. Then, search for the official Centos7 AMI (**ami-02676464a065c9c05**) and configure your instance using the recommended settings. We recommend the **c3.8xlarge** (32 cores and 60G of memory) or minimally **r4.2xlarge** (8 cores, 61G memory), which are both plenty for processing a full run. **We will also be attaching 500G** of storage which is roughly double the amount of space required for everything we're about to do here. Performance-wise, this pipeline is bottlenecked at two steps 1) mapping with STAR requires at least 32G for human assemblies, 2) peak calling with Clipper takes several hours using 16 cores but can scale up to as many as are available on your machine.

⚠ If you do not see the AMI, try changing the region to "N. California" (us-west-1). Otherwise, you may choose any Centos7-based image, which should not affect your results in any major way. We recommend a naked image with minimal bundled applications, which will minimize the risk of dependency issues.

Note: at some point you will need to set up your account credentials and create/download a .pem key. You may read the documentation from AWS here and will need to refer to this key when you login (see: Login!)





Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

ami-02676464a065c9c05

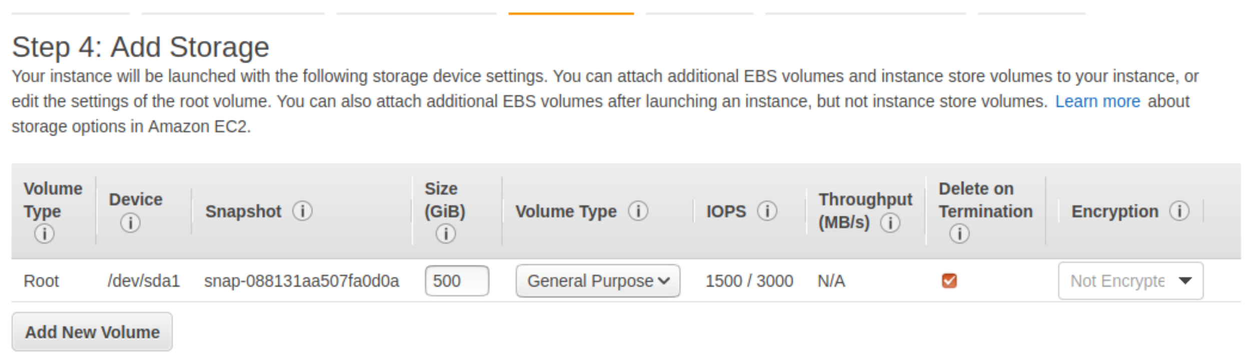
CentOS 7.8.2003 x86_64 - ami-02676464a065c9c05

CentOS 7.8.2003 x86_64

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

64-bit (x86)

We'll be using CentOS 7.8 although other flavors may be used.



Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

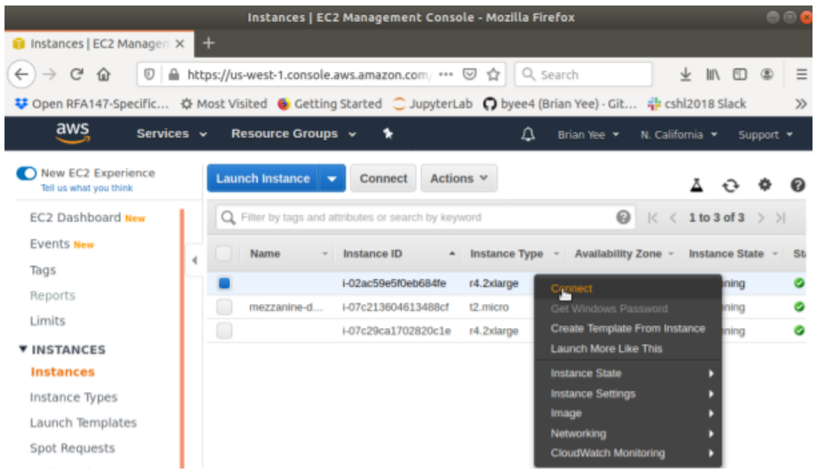
Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-088131aa507fa0d0a	500	General Purpose	1500 / 3000	N/A	✓	Not Encrypte

Add New Volume

(Optional) Setup a "Cost budget" on AWS to remind you of any unexpected charges or caps!

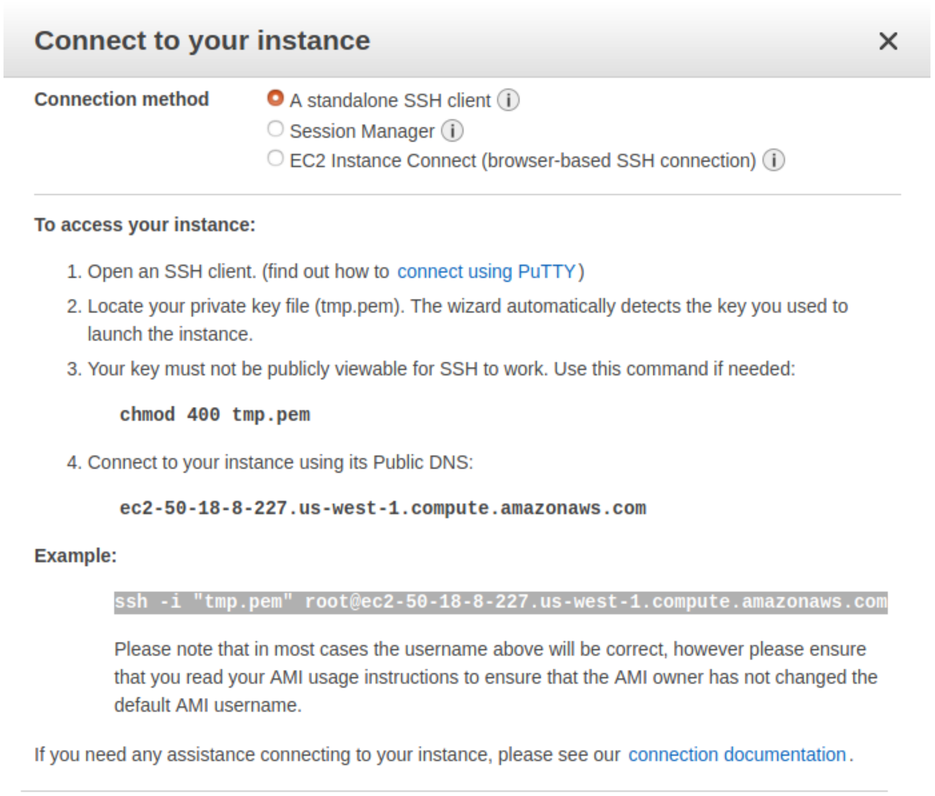
Login!

The easiest way to connect is to locate and right click your running instance to connect.



Name	Instance ID	Instance Type
	i-02ac59e5f0eb684fe	r4.2xlarge
mezzanine-d...	i-07c213604613488cf	t2.micro
	i-07c29ca1702820c1e	r4.2xlarge

From the console, right-click on your running instance and select 'connect'


Connect to your instance

Connection method: [✓ A standalone SSH client] Session Manager / EC2 Instance Connect (browser-based SSH connection)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (tmp.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 tmp.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-50-18-8-227.us-west-1.compute.amazonaws.com
```

Example:

```
ssh -i "tmp.pem" root@ec2-50-18-8-227.us-west-1.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

Open your terminal (Mac or Linux) and connect to your ec2 instance using the appropriate DNS and credentials. Windows users may need to download an ssh client (ie. PuTTY, git bash).

Note: You may need to change the example command to point to the correct location of your "pem" key, as well as login using "centos" instead of "root." So in this example, we would open our terminal and type:

`ssh -i "tmp.pem" centos@ec2-50-18-8-227.us-west-1.compute.amazonaws.com`

2. Setting up your initial environment

~0 - 5 minutes

Since we're starting from scratch, we will need to install some basic packages.

Update packages:
`sudo yum install -y update;`

Install the screen tool to preserve your session and wget for downloading:

`sudo yum install -y install screen wget;`

Install the C compiler (gcc):
`sudo yum -y group install "Development Tools" # installs gcc`

Install pip, python, and nodejs which are essential for CWL:

`wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh`

wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh

`sh Miniconda3-latest-Linux-x86_64.sh` (and follow onscreen instructions and click "yes" to initialize conda). Use defaults wherever prompted.

Re-run your `~/.bashrc` file and verify successful python/pip installation:

`source ~/.bashrc;`

`which pip;`

`which python;`

💡 Recent updates to pip has made it difficult to install certain packages (ruamel), so for the time being, please downgrade your pip to a version that works:

`pip install pip==20.1.1`

3. Install Docker and CWL

~0 - 5 minutes

Docker will help us pull pre-built containers with the correct environments and tools for each step of this pipeline. Most of these tools (such as samtools, bedtools, perl) are fairly straightforward on their own to install, but using these containers will simplify the tutorial.

We've elected to use a common workflow language (CWL) to define each step of our pipeline.

Install Docker:

Follow the instructions for installing Docker for CentOS. At the time of writing, the basic commands to install are as follows:

```
sudo yum install -y yum-utils
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install -y docker-ce docker-ce-cli containerd.io ### select "y"es to install Docker.
```

```
sudo systemctl start docker
```

⚠ Make sure you have started Docker with `sudo systemctl start docker` and verify installation with `sudo docker run hello-world`

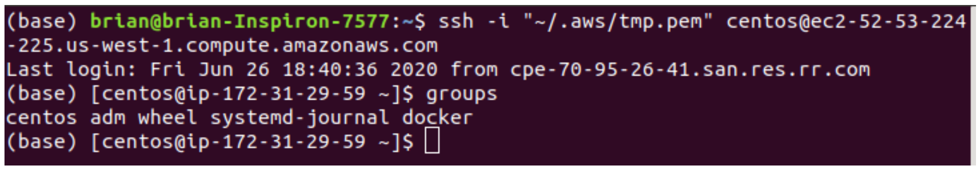
By default, this will install Docker however we will need to add our user (centos) to the docker group. So we must add ourselves to the group (docker):

`sudo groupadd docker`

`sudo usermod -aG docker ${USER}`

Log out and log back in to ensure you've been added to the `docker` group.

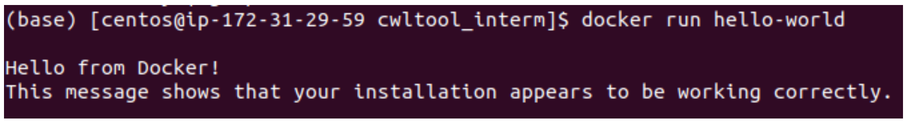
Verify that (centos) belongs to (docker) with `groups`:



```
(base) brian@brian-Inspiron-7577:~$ ssh -i "~/.aws/tmp.pem" centos@ec2-52-53-224
-225.us-west-1.compute.amazonaws.com
Last login: Fri Jun 26 18:40:36 2020 from cpe-70-95-26-41.san.res.rr.com
(base) [centos@ip-172-31-29-59 ~]$ groups
centos adm wheel systemd-journal docker
(base) [centos@ip-172-31-29-59 ~]$ 
```

Make sure your user belongs to the "docker" group.

And you should be able to run docker without sudo. Try it! `docker run hello-world`:



```
(base) [centos@ip-172-31-29-59 cwltool_interm]$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Install CWL:

~0 - 5 minutes

We can use pip to install cwl. Here we are using version (3.0.20200706173533).

`pip install cwltool==3.0.20200706173533`

You can verify installation with: `cwltool --help`

4. Download data and make references

~30 minutes - 1 hour

The following reference files are included in the eclip repository and

Download assembly fasta files and create the STAR indices:

Make an index for hg19 or hg38. Here we're using the assembly from Gencode and the STAR 2.7.6 Docker container:

```
cd ~
mkdir refs && cd refs
wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_34/GRCh37_mapping/GRCh37.primary_assembly.genome.fa.gz && \
gunzip GRCh37.primary_assembly.genome.fa.gz
mkdir STAR;

docker pull brianyee/star:2.7.6a
docker run -v /home/centos/refs:/refs brianyee/star:2.7.6a STAR \
--runThreadN 8 \
--runMode genomeGenerate \
--genomeDir refs/STAR \
--genomeFastaFiles refs/GRCh37.primary_assembly.genome.fa
```

💡 If you are aligning to GRCh38 (hg38), please use the assembly from encodeproject.org for this demo to ensure the chromosomes line up with Clipper's internal database

```
cd ~
mkdir refs && cd refs;
wget https://www.encodeproject.org/files/GRCh38_no_alt_analysis_set_GCA_000001405.15/@@download/GRCh38_no_alt_analysis_set_GCA_000001405.15
gunzip GRCh38_no_alt_analysis_set_GCA_000001405.15.fasta.gz
mkdir STAR;

docker pull brianyee/star:2.7.6a
docker run -v /home/centos/refs:/refs brianyee/star:2.7.6a STAR \
--runThreadN 8 \
--runMode genomeGenerate \
--genomeDir refs/STAR \
--genomeFastaFiles refs/GRCh38_no_alt_analysis_set_GCA_000001405.15.fasta
```

Make an index for repetitive sequences. You are free to make your own here, however we can provide an example for humans upon email request:

```
URL=https://external-collaborator-data.s3-us-west-1.amazonaws.com/reference-data/homo_sapiens_repbase_fixed_v2.fasta
cd ~
cd refs
wget $URL
mkdir STAR_repeat;
docker run -v /home/centos/refs:/refs brianyee/star:2.7.6a STAR \
--genomeSAindexNbases 9 \
--runThreadN 8 \
--runMode genomeGenerate \
--genomeDir refs/STAR_repeat \
--genomeFastaFiles refs/homo_sapiens_repbase_fixed_v2.fasta
```

Download example eCLIP data

~5 - 10 minutes

paired-end eCLIP

```
cd ~
wget https://external-collaborator-data.s3-us-west-1.amazonaws.com/reference-data/204_01_RBFOX2.tar.gz && tar -xvf 204_01_RBFOX2.tar.gz
```

single-end eCLIP

```
cd ~
wget https://external-collaborator-data.s3-us-west-1.amazonaws.com/reference-data/RBFOX2_INV.tar.gz && tar -xvf RBFOX2_INV.tar.gz
```

Download the eclip pipeline & add relevant directories to your `$PATH`

```
cd ~
git clone https://github.com/yeolab/eclip
export PATH=/home/centos/eclip/bin:/home/centos/eclip/cwl:/home/centos/eclip/wf:$PATH
```

5. Run the pipeline

~24-36 hours

Note: best to start a screen session - processing a sample can take up to a full day!

If you've made this far, you should be able to:

1. Verify that Docker and CWL are installed
2. Verify that you have the reference and example data downloaded
3. Verify that the pipeline files can be found in your `$PATH`

At this point, you should be able to navigate to the '`/home/centos/eclip/example`' directory and run either the `paired_end_clip.yaml` or `single_end_clip.yaml` manifests (the paths should be set up to point to existing files as outlined from this tutorial, but you may need to double check that the paths do exist)

`cd /home/centos/eclip/example`
`./paired_end_clip.yaml`

or

`./single_end_clip.yaml`

Troubleshooting:

The pipeline appears to run, but I see warnings.

Do not worry if you see a warning message at the beginning of your log file that looks like:

```
Source 'output_trim' of type ["null", {"type": "array",
  "items": "File"}] may be incompatible
  with sink 'input_fastqsort_fastq' of type {"type":
  "array", "items": "File"}
```

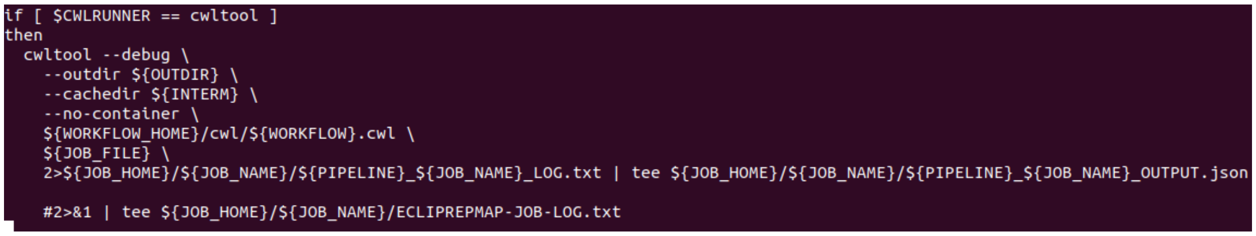
This is a silly side effect of trying to use one tool for both single-end and paired-end reads.

I'm getting an error: [Errno 2] No such file or directory

Since our own cluster disallows Docker (common among HPC due to permissions restrictions), it's possible that we may accidentally overwrite the parameter that runs Docker containers. When this happens, you may see your run fail immediately with a message telling you that the pipeline cannot find the tools: `ERROR 'eclipdemux' not found: [Errno 2] No such file or directory: 'eclipdemux': 'eclipdemux'`

How to fix:

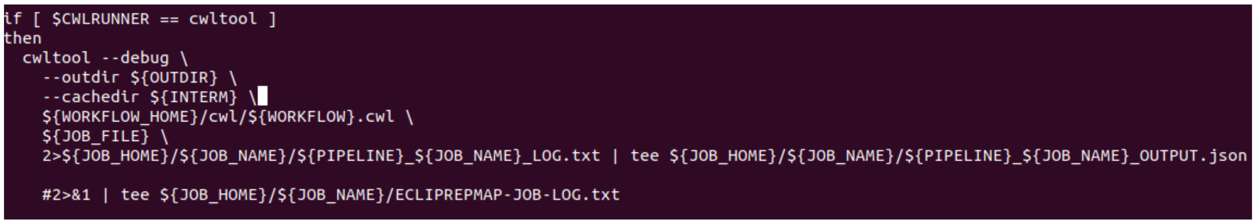
Fortunately, this is an easy fix. Using your favorite text editor, locate the cwl command within `eCLIP_singleend_singlenode` (for single-end processing) or `eCLIP_pairedend_singlenode` (for paired-end) and remove the `--no-container` parameter:



```
if [ $CWLRUNNER == cwltool ]
then
  cwltool --debug \
    --outdir ${OUTDIR} \
    --cachedir ${INTERM} \
    --no-container \
    ${WORKFLOW_HOME}/cwl/${WORKFLOW}.cwl \
    ${JOB_FILE} \
    2>${JOB_HOME}/${JOB_NAME}/${PIPELINE}_${JOB_NAME}_LOG.txt | tee ${JOB_HOME}/${JOB_NAME}/${PIPELINE}_${JOB_NAME}_OUTPUT.json

    #2>&1 | tee ${JOB_HOME}/${JOB_NAME}/ECLIPREPMAP-JOB-LOG.txt
```

Since this tutorial runs the pipeline with cwltool, you can scroll down to where the command is invoked.



```
if [ $CWLRUNNER == cwltool ]
then
  cwltool --debug \
    --outdir ${OUTDIR} \
    --cachedir ${INTERM} \
    ${WORKFLOW_HOME}/cwl/${WORKFLOW}.cwl \
    ${JOB_FILE} \
    2>${JOB_HOME}/${JOB_NAME}/${PIPELINE}_${JOB_NAME}_LOG.txt | tee ${JOB_HOME}/${JOB_NAME}/${PIPELINE}_${JOB_NAME}_OUTPUT.json

    #2>&1 | tee ${JOB_HOME}/${JOB_NAME}/ECLIPREPMAP-JOB-LOG.txt
```

Removing the ```--no-container``` parameter will enable pulls from Docker.

No mandatory DockerRequirement Error:

```
ERROR Workflow error:
No mandatory DockerRequirement, yet path is outside the designated output directory, also know as $(runtime.outdir): MapperEnt(resolved='/h
```

We've noticed that cwltool versions≥**3.0.20200807132242** is stricter about its Docker requirement (our pipelines suggest "hints" but does not require docker since some clusters (including our own) do not have the capability of running Docker.

How to fix:

Option 1. Install a previous version of cwltool (ie. `pip install cwltool==3.0.20200706173533`)

Option 2. Modify the tool definition (`.cwl`) files to make Docker mandatory. Below is an example using the first 20 lines of the `eclip/cwl/makebigwigfiles_PE.cwl` tool. You will simply need to move the Docker definition from `hints` to `requirements`, however be careful to preserve spacing, as these files should be of YAML format (**make sure the ticks line up**):

```
#!/usr/bin/env cwltool

cwlVersion: v1.0

class: CommandLineTool

requirements:
```

```
  - class: ResourceRequirement
    coresMin: 1
    ramMin: 8000
  - class: InitialWorkDirRequirement
    listing:
      - entry: $(inputs.bam)
        writable: true

hints:
  - class: DockerRequirement
    dockerPull: brianyee/makebigwigfiles:0.0.3

baseCommand: [makebigwigfiles]
```

```
#!/usr/bin/env cwltool

cwlVersion: v1.0

class: CommandLineTool

requirements:
  - class: ResourceRequirement
    coresMin: 1
    ramMin: 8000
  - class: InitialWorkDirRequirement
    listing:
      - entry: $(inputs.bam)
        writable: true
  - class: DockerRequirement
    dockerPull: brianyee/makebigwigfiles:0.0.3

baseCommand: [makebigwigfiles]
```

Typing "cwltool -h" gives the following error: "ModuleNotFoundError": "No module named 'ruamel'"

See the above callout which provides one workaround, which is to downgrade pip:
`pip install pip==20.1.1`

References:

Van Nostrand, Eric L., et al. "Principles of RNA processing from analysis of enhanced CLIP maps for 150 RNA binding proteins." Genome biology 21 (2020): 1-26.

Van Nostrand, Eric L., et al. "A large-scale binding and functional map of human RNA-binding proteins." Nature 583.7818 (2020): 711-719.