Working with the AWS Job Runner

The JobRunner is a java cmd line tool for running bash scripts on AWS EC2 nodes, see https://github.com/HuntsmanCancerInstitute/AwsApps

JobRunner Menu

```
 u0028003$ java -jar -Xmx1G ~/Code/AwsApps/target/JobRunner_0.2.jar

**********************************************************************************************************************
**                                          AWS Job Runner : December 2021                                            **
**********************************************************************************************************************
JR is an app for running bash scripts on AWS EC2 nodes. It downloads and uncompressed your resource bundle and looks for
xxx.sh_JR_START files in your S3 Jobs directories. For each, it copies over the directory contents, executes the
associated xxx.sh script, and transfers back the results.  This is repeated until no unrun jobs are found. Launch many
EC2 JR nodes, each running an instance of the JR, to process hundreds of jobs in parallel. Use spot requests and
hibernation to reduce costs.

To use:
1) Install and configure the aws cli on your local workstation, see https://aws.amazon.com/cli/
2) Upload your aws credentials file into a private bucket on aws, e.g.
     aws s3 cp ~/.aws/credentials s3://my-jr/aws.cred.txt
3) Generate a secure 24hr timed URL for the credentials file, e.g.
     aws --region us-west-2  s3 presign s3://my-jr/aws.cred.txt  --expires-in 259200
4) Upload a zip archive containing resources needed to run your jobs into S3, e.g.
     aws s3 cp ~/TNRunnerResourceBundle.zip s3://my-jr/TNRunnerResourceBundle.zip
     This will be copied into the /JRDir/ directory and then unzipped.
5) Upload script and job files into a 'Jobs' directory on S3, e.g.
     aws s3 cp ~/JRJobs/A/ s3://my-jr/Jobs/A/ --recursive
6) Optional, upload bash script files ending with JR_INIT.sh and or JR_TERM.sh. These are executed by JR before and after
     running the main bash script.  Use these to copy in sample specific resources, e.g. fastq/ cram/ bam files, and to run
     post job clean up.
7) Upload a file named XXX_JR_START to let the JobRunner know the bash script named XXX is ready to run, e.g.
     aws s3 cp s3://my-jr/emptyFile s3://my-jr/Jobs/A/dnaAlignQC.sh_JR_START
8) Launch the JobRunner.jar on one or more JR configured EC2 nodes. See https://ri-confluence.hci.utah.edu/x/gYCgBw

Job Runner Options:
-c URL to your secure timed config credentials file.
-r S3URI to your zipped resource bundle.
-j S3URI to your root Jobs directory containing folders with job scripts to execute.
-l S3URI to your Log folder for node logs.

Default Options:
-d Directory on the local worker node, full path, in which resources and job files will be processed, defaults to /JRDir/
-a Aws credentials directory, defaults to ~/.aws/
-t Terminate the EC2 node upon job completion. Defaults to looking for jobs for the min2Wait.
-w Minutes to wait when jobs are not found before termination, defaults to 10.
-x Replace S3 job directories with processed analysis, defaults to syncing local with S3. WARNING, if selected, don't place
     any files in these S3 jobs directories that cannot be replaced. JR will delete them.
-v Verbose debugging output.

Example: java -jar -Xmx1G JobRunner.jar
     -r s3://my-jr/TNRunnerResourceBundle.zip
     -j s3://my-jr/Jobs/
     -l s3://my-jr/NodeLogs/
     -c 'https://my-jr.s3.us-west-2.amazonaws.com/aws.cred.txt?X-Amz-Algorithm=AWS4-HMXXX...'
```

****

To configure EC2 nodes for the JobRunner:

1) **Log into** the AWS Console in the numbered account you would like to bill https://console.aws.amazon.com/console/home?nc2=h_ct&src=header-signin

2) **Navigate** to the EC2 service, https://us-west-2.console.aws.amazon.com/ec2

3) **Check** that the region for EC2 is the same as where your data is stored, top menu bar → 2nd drop down from right next to your name. This is important so that you don't incur data egress fees. Within region transfers are free.

4) **Click** the big orange *Launch instances* button on the top right.

5) **Select** the top Amazon Linux 2 AMI starting image to load, this is the first in the list, use the default settings.

6) Now find an appropriate server type to run. This should be ESB only, ESB optimized, many available in your region for spot pricing, and support hibernation. For example, under the **All instance families drop down**, select c5. Then **select the checkbox** next to the c5.9xlarge (36 cpu, 72 gb). This is a good general use server. There are hundreds of other options. **Click** the *Next: Configure Instance Details*.

7) Lots of switches and dials to tune your servers, these three are the key to set:

- **Number of instances** : keep this small to start, this is the number of individual JobRunner servers to start
- **Purchasing option** : select Request Spot instances → click the Persistent request → set the Interruption behavior to Hibernate
- **User data**: paste in bash cmd lines to execute upon server startup, e.g. for JobRunner

User data

```
#!/bin/bash
sudo su root

# Make a directory in which to run the JobRunner app
mkdir /JRDir; chmod -R 777 /JRDir; cd /JRDir

# Update the AMI and install a variety of packages
yum -y update &> ec2NodeStartup.log
yum install -y rpm-build wget &>> ec2NodeStartup.log
yum groupinstall -y 'Development Tools' &>> ec2NodeStartup.log
amazon-linux-extras install java-openjdk11 epel -y &>> ec2NodeStartup.log

# Install Singularity
yum install -y golang libseccomp-devel squashfs-tools cryptsetup  &>> ec2NodeStartup.log
wget http://bioserver.hci.utah.edu/NixMisc/singularity-3.8.0-1.amzn2.x86_64.rpm &>> ec2NodeStartup.log
echo -e "\nRPMingSingularity" &>> ec2NodeStartup.log
rpm -ivh singularity-3.8.0-1.amzn2.x86_64.rpm &>> ec2NodeStartup.log
rm -f singularity-3.8.0-1.amzn2.x86_64.rpm &>> ec2NodeStartup.log

# Enable hiberation
/usr/bin/enable-ec2-spot-hibernation &>> ec2NodeStartup.log

# Download the latest JobRunner, check path at https://github.com/HuntsmanCancerInstitute/AwsApps/releases
wget https://github.com/HuntsmanCancerInstitute/AwsApps/releases/download/JobRunner_0.2/JobRunner_0.2.jar &>> ec2NodeStartup.log
```

```
# Check apps
echo -e "\nSingularity and Java Checks:" &>> ec2NodeStartup.log
singularity version  &>> ec2NodeStartup.log && \
java -version  &>> ec2NodeStartup.log && \
echo READY &>> ec2NodeStartup.log

# On your local workstation, pull a timed URL to your aws credentials file and add it here, see the JobRunner.jar menu, e.g. 'aws --region us-west-2  s3 presign s3://my-
jr/aws.cred.txt  --expires-in 259200'
credUrl="https://hcibioinfo-jobrunner.s3.us-west-2.amazonaws.com/Credentials/aws.cred.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIARHBYAZBR33RCJK6A%2F20211122%2Fus-
west-2%2Fs3%2Faws4_request&X-Amz-Date=20211122T232159Z&X-Amz-Expires=259200&X-Amz-SignedHeaders=host&X-Amz-
Signature=a699fb83414fb604455b55c76eb51ece8b4be60d356c5c0109cdc3cd1ce76a5d"

# Launch the JobRunner in termination mode -t to shut down the EC2 node upon all job completion
java -jar -Xmx1G JobRunner_0.1.jar -c $credUrl \
-r s3://hcibioinfo-jobrunner/ResourceBundles/dnaAlignQC_3Nov2021.zip \
-j s3://hcibioinfo-jobrunner/Jobs/ \
-l s3://hcibioinfo-jobrunner/NodeLogs/ -t -x &> jr.out &
```

- **Click** Next: Add Storage

8) Increase the Size of the EBS storage volume, this needs to be big enough for all of your resource and job data files as well as some overhead for hibernation system files, 250 is good for exome alignment. Encrypt the volume by selecting the orange default option. Don't add a New Volume, just modify the default. **Click** Next: Add Tags

9) Add tags, a good one is Name: YourName, so you know which rentals are yours. **Click** Review and Launch

10) Check that everything looks as expected, then **click** Launch

11) Go back to the EC2 dashboard, **click** Instances from the left side bar to see the status of each of your nodes, connect to one of them by **selecting** a row check box -> **clicking** the big Connect button then → EC2 Instance Connect → big orange Connect button to launch a cmd line terminal in your web browser. Change into the /JRDir and monitor the ec2NodeStartup.log and then the jr.out file to watch your jobs run.

12) While your jobs are running **cancel the persistent spot requests**! Otherwise at some point JR will look for jobs, find none, and then shutdown the node. EC2 will then relaunch a new node. Stop this by **going to** the EC2 dashboard → **click** the Spot Requests from the left sidebar → **click** the all spot request check box → **select** Cancel request under the Actions dropdown. In the popup uncheck Terminate instances checkbox to leave the existing nodes running.

13) On your local workstation in a terminal, monitor your jobs by listing the file contents, e.g. 'aws s3 ls s3://hcibioinfo-jobrunner/Jobs --recursive'. Running jobs will contain a file labeled scriptName_JR_RUNNING_hostID. Completed jobs will contain a log file named scriptName_JR_LOG.txt or scriptName_JR_LOG_ERROR.txt if any of your bash scripts had a non zero exit status.

14) To shutdown the EC2 nodes kill the spot requests first if you haven't done so, see 12). **Go to** the EC2 dashboard → **click** Instances from the left sidebar → **click** the check boxes → **select** Terminate instance under the Instance state dropdown. Lastly, **check** the Instances and Spot Requests status to make sure all are shutting down or Terminated.

A HaplotypeCalling Example:

HaplotypeCalling

```
# Upload cram files to aws for haplotype calling

## create a directory of linked cram files
cd /uufs/chpc.utah.edu/common/HIPAA/u0028003/Scratch/Gta/ARID1A/ForAws
for x in $(realpath ../Arid1aJobs/*/Alignment/*_NormalDNA/Alignment/*cram*); do ln -s $x .; done

## upload them in screen
```

```
cd /uufs/chpc.utah.edu/common/HIPAA/u0028003/Scratch/Gta/ARID1A
module load python/3.9.7
        ### dry run
        aws s3 cp --dryrun --recursive --follow-symlinks ForAws/ s3://hcibioinfo-jobrunner/Gta/GATKJobs/NormalCrams/ && echo SUCCESS || echo FAIL
        ### for real
        aws s3 cp --recursive --follow-symlinks ForAws/ s3://hcibioinfo-jobrunner/Gta/GATKJobs/NormalCrams/ && echo SUCCESS || echo FAIL


# clean out any prior jobs on s3
aws s3 rm --recursive s3://hcibioinfo-jobrunner/Gta/GATKJobs/Haplo/

# Make the haplo jobs
cd /uufs/chpc.utah.edu/common/HIPAA/u0028003/Scratch/Gta/ARID1A/AwsJobs/Haplo
## modify the yaml for aws
cp ~/TNRunner/Workflows/UpdatedAug2021/tnRunner.yaml haplotypeCalling.yaml
nano haplotypeCalling.yaml

for x in YPHY3TK5DX-IDT-SL407275-SL406988-SL414867 Z5SSPJMZU7-NIM-SL344402-SL347188-NA ZLQBHNC7JH-IDT-SL434732-SL434913-NA ...
do
        echo; echo $x
        d=$x/GermlineVariantCalling/$x"_GVCF"
        mkdir -p $d

        # Add the tnRunner script files
        cp ~/TNRunner/Workflows/UpdatedAug2021/GATKGermline/HaplotypeCalling/haplotypeCalling.* $d/
        cp /scratch/general/pe-nfs1/u0028003/Gta/ARID1A/AwsJobs/Haplo/haplotypeCalling.yaml $d/

        # Add an init file
        echo "echo Downloading the alignment cram..." > $d/JR_INIT.sh
        echo "aws s3 cp --quiet s3://hcibioinfo-jobrunner/Gta/GATKJobs/NormalCrams/"$x"_NormalDNA_Hg38.cram ." >> $d/JR_INIT.sh
        echo "aws s3 cp --quiet s3://hcibioinfo-jobrunner/Gta/GATKJobs/NormalCrams/"$x"_NormalDNA_Hg38.cram.crai ." >> $d/JR_INIT.sh

        # make term file
        echo "echo Deleting the cram..." > $d/JR_TERM.sh
        echo "rm -rf *cram*" >> $d/JR_TERM.sh
        echo "mv -f JR_INIT.sh RunScripts/ " >> $d/JR_TERM.sh
        echo "mv -f JR_TERM.sh RunScripts/ " >> $d/JR_TERM.sh

        # add a start file
        touch $d/haplotypeCalling.README.sh"_JR_START"
done

# copy the job dirs to s3
cd /uufs/chpc.utah.edu/common/HIPAA/u0028003/Scratch/Gta/ARID1A/AwsJobs
module load python/3.9.7
## dry run
aws s3 cp --dryrun --recursive --follow-symlinks Haplo/ s3://hcibioinfo-jobrunner/Gta/GATKJobs/Haplo/ && echo SUCCESS || echo FAIL
## for real, this will replace any existing files in those dirs
aws s3 cp --recursive --follow-symlinks Haplo/ s3://hcibioinfo-jobrunner/Gta/GATKJobs/Haplo/ && echo SUCCESS || echo FAIL
## checkem
aws s3 ls --recursive s3://hcibioinfo-jobrunner/Gta/GATKJobs/Haplo/

# make a zip bundle of the resources needed for the HaplotypeCaller workflow and upload it into the resource folder on s3
cd /uufs/chpc.utah.edu/common/PE/hci-bioinformatics1
zip -r haplotypeCalling_3Dec2021.zip  \
   TNRunner/GATKResourceBundleAug2021/Homo_sapiens_assembly38.fasta \
   TNRunner/GATKResourceBundleAug2021/Homo_sapiens_assembly38.dict \
```

```
   TNRunner/GATKResourceBundleAug2021/Homo_sapiens_assembly38.fasta.fai \
   TNRunner/Containers/public_GATK_SM_1.sif \
   TNRunner/Bed/AvatarMergedNimIdtBeds/hg38NimIdtMergedPad150bp.bed.gz* \
   TNRunner/GATKResourceBundleAug2021/dbsnp_146.hg38.vcf.gz* \
   TNRunner/GATKResourceBundleAug2021/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz* \
   TNRunner/Vcfs/Homo_sapiens_assembly38.known_indels.vcf.gz* \
   TNRunner/GATKResourceBundleAug2021/1000G_phase1.snps.high_confidence.hg38.vcf.gz* && echo OK || echo FAIL
aws s3 cp haplotypeCalling_3Dec2021.zip s3://hcibioinfo-jobrunner/ResourceBundles/

# purge node logs
aws s3 rm s3://hcibioinfo-jobrunner/NodeLogs/ --recursive

# pull a timed url to the credentials file, save the URL for the EC2 user data
aws --region us-west-2 s3 presign s3://hcibioinfo-jobrunner/Credentials/aws.cred.txt  --expires-in 259200

# launch one or more JR EC2 instances following the User Data example above, e.g. :
#!/bin/bash
set -e
sudo su root
# Make a directory in which to run the JobRunner app
mkdir /JRDir; chmod -R 777 /JRDir; cd /JRDir
# Update the AMI and install a variety of packages
yum -y update &> ec2NodeStartup.log
yum install -y rpm-build wget &>> ec2NodeStartup.log
yum groupinstall -y 'Development Tools' &>> ec2NodeStartup.log
amazon-linux-extras install java-openjdk11 epel -y &>> ec2NodeStartup.log
# Install Singularity
yum install -y golang libseccomp-devel squashfs-tools cryptsetup  &>> ec2NodeStartup.log
wget http://bioserver.hci.utah.edu/NixMisc/singularity-3.8.0-1.amzn2.x86_64.rpm &>> ec2NodeStartup.log
echo -e "\nRPMingSingularity" &>> ec2NodeStartup.log
rpm -ivh singularity-3.8.0-1.amzn2.x86_64.rpm &>> ec2NodeStartup.log
rm -f singularity-3.8.0-1.amzn2.x86_64.rpm &>> ec2NodeStartup.log
# Enable hiberation
/usr/bin/enable-ec2-spot-hibernation &>> ec2NodeStartup.log
# Download the latest JobRunner
wget https://github.com/HuntsmanCancerInstitute/AwsApps/releases/download/JobRunner_0.2/JobRunner_0.2.jar &>> ec2NodeStartup.log
# Check apps
echo -e "\nSingularity and Java Checks:" &>> ec2NodeStartup.log
singularity version  &>> ec2NodeStartup.log && \
java -version  &>> ec2NodeStartup.log && \
echo READY &>> ec2NodeStartup.log
# Launch the JobRunner
credUrl="https://hcibioinfo-
jobrunner.s3.amazonaws.com/Credentials/aws.cred.txt?AWSAccessKeyId=AKIARHBYAZBR33RCJK6A&Expires=1639415807&Signature=KGiiei%2FAil9fKVspT1tvvOH%2BhoY%3D"
nohup java -jar -Xmx1G JobRunner_0.2.beta.jar -c $credUrl \
-r s3://hcibioinfo-jobrunner/ResourceBundles/haplotypeCalling_3Dec2021.zip \
-j s3://hcibioinfo-jobrunner/Gta/GATKJobs/Haplo/ \
-l s3://hcibioinfo-jobrunner/NodeLogs/ -x -t &> jr.out &

# Monitor the jobs by watching for log files
aws s3 ls --recursive s3://hcibioinfo-jobrunner/Gta/GATKJobs/Haplo/ | grep JR_LOG

# Use the AWS Console to ssh into a running instance and watch the logs in /JRDir/, monitor cpu and memory usage with top.
```