Application of Singularity Containers

Background

We wanted a pipeline calling low level mosaic variants in patients with overgrowth disorders.

Tools to be installed

Tool	Purpose
BWA	Read alignment
Samtools Suite	BAM file preprocessing
Picard	BAM file preprocessing
GATK	BAM file preprocessing
LoFreq	Variant calling

Recipe file

```
$ cat mosaic

Bootstrap: debootstrap
OSVersion: xenial
MirrorURL: http://us.archive.ubuntu.com/ubuntu

%runscript
    echo "mosaicpipeline v. 1.0"

%post
    #install dependencies
    apt-get update
    apt-get install -y build-essential
    apt-get install -y zlib1g-dev
    apt-get install -y locales
    locale-gen en_US
    apt-get update
    apt-get install -y git
    cd /usr/bin

    apt-get update
    apt-get install -y gcc
    apt-get install -y make
    apt-get install -y libbz2-dev
    apt-get install -y libncurses5-dev
    apt-get install -y libncursesw5-dev
    apt-get install -y liblzma-dev
    apt-get install -y wget
    apt-get install -y python
    cd /usr/bin

    #install bwa
```

```
    apt-get -y install unzip
    unzip bwa-0.7.17.zip
    cd bwa-0.7.17; make; make install
    cd /usr/bin

    #install htslib
    wget https://github.com/samtools/htslib/releases/download/1.9/htslib-1.9.tar.bz2
    tar -vxjf htslib-1.9.tar.bz2
    cd htslib-1.9
    make
    make install

    #install samtools
    cd ..
    wget https://github.com/samtools/samtools/releases/download/1.9/samtools-1.9.tar.bz2
    tar -vxjf samtools-1.9.tar.bz2
    cd samtools-1.9
    make
    make install

    #install bcftools
    cd ..
    wget https://github.com/samtools/bcftools/releases/download/1.9/bcftools-1.9.tar.bz2
    tar -vxjf bcftools-1.9.tar.bz2
    cd bcftools-1.9
    make
    make install

    #install JAVA
    cd /usr/bin
    tar -xzvf jdk-8u212-linux-x64.tar.gz
    export JAVA_HOME="/usr/bin/jdk1.8.0_212"

    #install picard
    cd /usr/bin
    git clone --branch 2.18.26 --depth 1 https://www.github.com/broadinstitute/picard.git
    cd picard
    ./gradlew shadowJar

    #install GATK
    cd /usr/bin
    tar -xjf GenomeAnalysisTK-3.8-1-0-gf15c1c3ef.tar.bz2

    #install lofreq
    tar -xzvf lofreq_star-2.1.3.1_linux-x86-64.tgz
    cp -rv ./lofreq_star-2.1.3.1/* /usr/local/
    chmod 755 /root
    chmod 755 /home
    chmod 755 /home/*

%environment
    export PATH="$PATH:/usr/bin/bcftools-1.9"
    export PATH="$PATH:/usr/bin/samtools-1.9"
```

```
    export PATH="$PATH:/usr/bin/htslib-1.9"
    export PATH="$PATH:/usr/bin/bwa-0.7.17/"
    export PATH="$PATH:/usr/local"
    export JAVA_HOME="/usr/bin/jdk1.8.0_212"
    export PATH=$JAVA_HOME/bin:$PATH

%files
    GenomeAnalysisTK-3.8-1-0-gf15c1c3ef.tar.bz2 /usr/bin
    lofreq_star-2.1.3.1_linux-x86-64.tgz /usr/bin
    jdk-8u212-linux-x64.tar.gz /usr/bin
    CLIA_scripts/* /home
    bwa-0.7.17.zip /usr/bin

%help

Mosaic Variant CLIA Pipeline SOP: mosaicpipeline v. 1.0

Pipeline Description
This pipeline is meant to call low level mosaic variants on samples sequenced on the Biesecker Lab\'s M

#PROGRAMS
BWA v. 0.7.17
SAMTOOLS v. 1.9
BCFTOOLS v. 1.9
PICARD v. 2.18.26
GATK v. 3.8-1-0
lofreq v. 2.1.3.1

1. Preparing Input
Before doing scp of fastq.gz files into biowulf, create a directory in
/data/BieseckerBioinfo/lofreq_pipeline/

Directory naming convention: YEAR_MONTH_DATE
Fastq files go into this new directory.

Inside the newly made directory, create a directory named out.
/data/BieseckerBioinfo/lofreq_pipeline/YEAR_MONTH_DATE/out

2. Instructions on how to run the mosaicpipeline v.1.0 container on biowulf:
Execute the wrapper script found in /data/BieseckerBioinfo/lofreq_pipeline/scripts:

    bash wrap.sh -d [directory which your fastq files are stored]

If you need to reproduce the wrapper script, you can get it from the singularity container with:
    singularity exec mosaicpipeline.simg cat /home/wrap.sh > wrap.sh

The scripts that were executed, log file, and output should be found in the out directory.

3. Output files and deliverables
A VCF file is produced per pair of fastq files in the output folder above. VCF file will then
get annotated using VEP and Gemini (not part of the container).

#CONTACT
```

```
Any questions or concerns please contact Henoke Shiferaw at henoke.shiferaw@nih.gov.
```

Building our Singularity container from Recipe

```
$ sudo singularity build mosaicpipeline.simg mosaic
```

Viewing the help section

Some people using your container might not know what is installed, what your container is made for, or how to run it. The %help section of your recipe file is a great place to give this information.

```
$ singularity help mosaicpipeline.simg
```

Reproducing your recipe file from your built container

Once built, the recipe file can easily be reproduced by the following command:

```
$ singularity inspect --deffile mosaicpipeline.simg
```

Sample Lofreq script

Below is a sample of the final step in my pipeline. I wanted to run each step of the container in parallel by sample.

You can see each line is an exec command that calls lofreq from my container.

Notice *"/home/custompool.bed"* is from my container. I added this file in the %file section of my container when I copied the CLIA_scripts files. Therefore, the only directory I had to bind here is to get the input.

Also, using *–bind* is not permanent and it is specific to the singularity command I am running. Therefore I have to spencify the directory I want to bind each time I run exec, even if it is the same directory.

```
singularity exec --bind /data/henoke/out/ mosaicpipeline.simg lofreq call -l /home/custompool.bed --cal
singularity exec --bind /data/henoke/out/ mosaicpipeline.simg lofreq call -l /home/custompool.bed --cal
singularity exec --bind /data/henoke/out/ mosaicpipeline.simg lofreq call -l /home/custompool.bed --cal
```

Working interactively inside the container

Using *singularity shell* allows us to enter the container and open up an interactive shell allowing you to call the tools you installed in your container as if they were on your host.

For example:

Since Samtools is installed in this container, let's try to index a bam file using samtools installed inside our container using singularity shell. Our bam file is in a different directory so we still must bind the directory when writing the command.

You will know you're inside the container when

```
$ singularity shell --bind /data/henoke/sandbox/ mosaicpipeline.simg
Singularity mosaicpipeline.simg:/data/henoke/scripts>
```

You'll know you're inside the container when the shell shows the container you're in the terminal prompt.

Now that we have entered the container. All commands are executed from the container, in the host filespace.

Now the syntax of indexing a bam file doesn't require 'exec'. You just type the command as you would normally

```
Singularity mosaicpipeline.simg:/data/henoke/scripts> samtools index /data/henoke/sandbox/a.bam
```

To exit interactive mode, just type exit.

```
Singularity mosaicpipeline.simg:/data/henoke/scripts> exit
```