Singularity

Singulariy installation

Following the singularity website gives some problems with go, is better this approach:

GO 1.18 is working correctly when it was written, but is possible changing the VERSION variable

```
sudo apt-get update && \
sudo apt-get install -y build-essential \
libseccomp-dev pkg-config squashfs-tools cryptsetup
sudo rm -r /usr/local/go
export VERSION=1.18 OS=linux ARCH=amd64  # change this as you need
wget -O /tmp/go${VERSION}.${OS}-${ARCH}.tar.gz https://dl.google.com/go/go${VERSION}.${OS}-
${ARCH}.tar.gz && \
sudo tar -C /usr/local -xzf /tmp/go${VERSION}.${OS}-${ARCH}.tar.gz
echo 'export GOPATH=${HOME}/go' >> ~/.bashrc && \
echo 'export PATH=/usr/local/go/bin:${PATH}:${GOPATH}/bin' >> ~/.bashrc && \
source ~/.bashrc
curl -sfL https://install.goreleaser.com/github.com/golangci/golangci-lint.sh |
sh -s -- -b $(go env GOPATH)/bin v1.21.0
mkdir -p ${GOPATH}/src/github.com/sylabs && \
cd ${GOPATH}/src/github.com/sylabs && \
git clone https://github.com/sylabs/singularity.git && \
cd singularity
git checkout v3.6.3
cd ${GOPATH}/src/github.com/sylabs/singularity && \
./mconfig && \
cd ./builddir && \
make && \
sudo make install
```

View is singularity is working:

```
singularity version
```

Singularity sandbox

To create a singularity container is very useful first start with a sandbox to view what is needed to install, run all the process and then create the image

Create first the sandbox (image where we will try things):

```
sudo singularity build --sandbox insurveyor library://ubuntu:22.04
```

If the base image come from docker: docker://kubor/giggle-docker Enter to the sandbox created:

```
sudo singularity shell --writable insurveyor
```

Inside the sandbox is possible to try all the things that next will introduce to .deb file (file where all the instructions for creating the image are placed)

Singularity build image (sif)

Once we have defined all processes needed and tested them we can create the image. For creating a sif image first is necessary create a deb file which will define the package and files to install en execute in the singularity image, which looks like this:

```
Bootstrap: library
From: ubuntu:22.04

%post
        apt-get update && apt-get install -y \
        autoconf \
        bzip2 \
        gcc \
        g++ \
        git \
        libbz2-dev \
        libcurl4-openssl-dev \
        libssl-dev \
        liblzma-dev \
        make \
        wget \
        zlib1g-dev \
        cmake

        apt install -y python3.8
        apt-get install -y python3-pip



        pip3 install numpy
        pip3 install pyfaidx
        pip3 install pysam

        cd /opt
        git clone https://github.com/kensung-lab/INSurVeyor
        cd INSurVeyor/
        ./build_htslib.sh
        cmake -DCMAKE_BUILD_TYPE=Release . && make

%environment
        export PATH=/opt/INSurVeyor:$PATH
```

First it is necessary to define desired bootstrap (docker) and distro (ubuntu:22.04).

In post will define the packages needed, using `-y` in `apt-get` for define as 'yes'. It is recommended to put the installations in /opt for avoiding root problems

files and runscript define the files and the different actions than will be run if singularity run is executed

To create the container from the recipe run:

```
sudo singularity build insurveyor.sif insurveyor.def
```

To enter interactively into the singularity image to try the installation parameters and try the different software installed.

In this case if we run:

```
singularity run insurveyor.sif
```

The surveyor.py python script will be run.

Is it possible then define the different arguments of the script

```
singularity run insurveyor.sif data/sample.bam data/sample_workdir data/sample.vcf
```

It is also possible to exec some code inside of the container;

```
singularity exec egs.sif
```

If the bin is defined in /usr/bin using

```
ln -s /opt/giggle/bin/giggle /usr/bin/
```

It will be possible to run:

```
singularity exec egs.sif giggle
```

It is possible to create different apps inside the singularity

```
%appenv rscript
    # Run an R script
    Rscript $@

%appenv pythonscript
    # Run a Python script
    python3 $@
```

And run as:

```
singularity run --app rscript egs.sif
singularity run --app pythonscript egs.sif
```

It is possible to export path defining them in export

```
%environment
# Export the PATH variable
export PATH=/my/custom/bin:$PATH
```

And then execute as:

```
singularity exec egs.sif
```