NEXTFLOW DOCUMENTATION

Docker installation in own computer

Docker documentation:
https://docs.docker.com/get-started/overview/

1- Install docker following the instructions in this link:
https://docs.docker.com/desktop/mac/install/

Once you install Docker in your computer you can log in using ciibioinformatics.

In the same way you can sing in in Docker Hub where all the images will be store to share.

https://hub.docker.com

username:
ciibioinformatics

password:
…….

Nextflow installation

Nextflow documentation:
https://www.nextflow.io/docs/latest/index.html

1- Install nextflow in your own computer

```
curl -s https://get.nextflow.io | bash
```

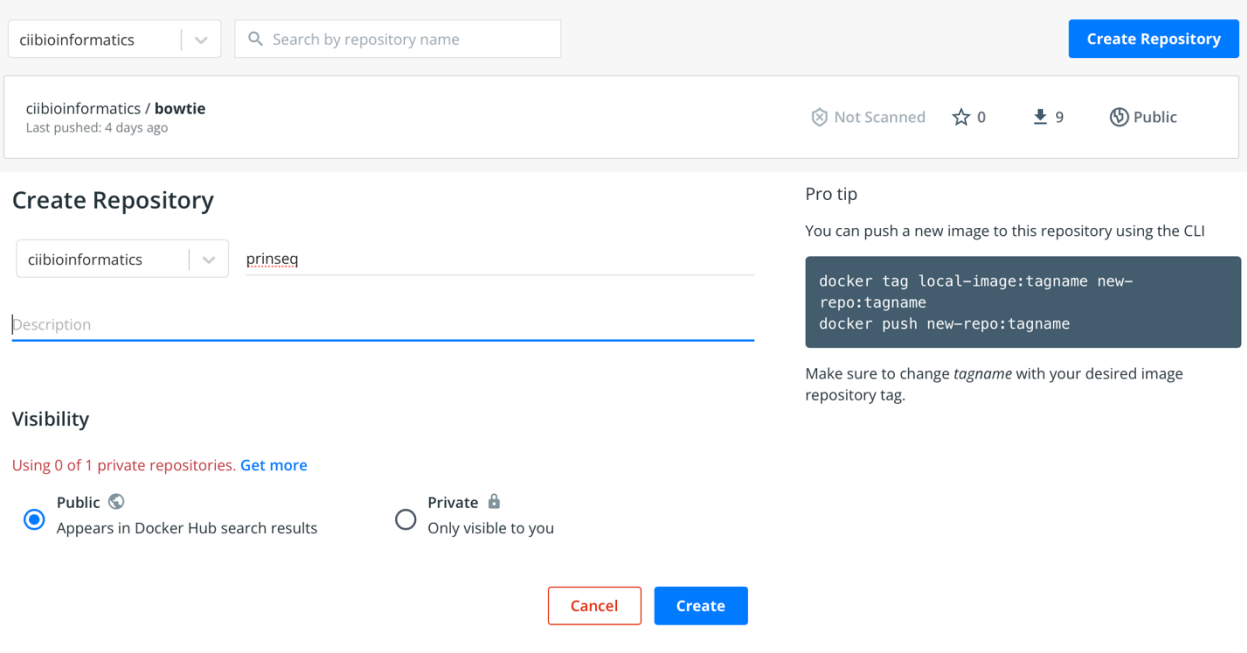
1- Export path so you don't have to type every time the whole path.

```
export PATH=$PATH:/place/with/the/file
```

Additional documentation about Nextflow:
https://nf-co.re
https://dockstore.org

DOCKER USAGE

We log into ciibioinformatics Docker Hub and create a repository:



Then, in our local computer we need to first create a Dockerfile (name of dockerfile has to be "Dockerfile"). Dockerfile will be something similar to this:

Get Started | conda_environment.sh | fastq_cuadapt_bowtie3.nf | cutadapt.3.0

sers > MAIDER > Documents > nextflow > docker_images > software_dockerfiles > cutadapt.3.0_Do

```
1   ############################################################
2   # Dockerfile to build Cutadapt container images
3   # Based on Ubuntu
4   ############################################################
5
6   # Set the base image to Ubuntu
7   FROM ubuntu:20.04
8
9   # File Author / Maintainer
10  MAINTAINER maider astorkia amiama, maa2360@cumc.columbia.edu
11
12  ARG PACKAGE_VERSION=3.0
13  ARG BUILD_PACKAGES="build-essential"
14  ARG DEBIAN_FRONTEND=noninteractive
15
16  RUN apt update && \
17      apt install --yes $BUILD_PACKAGES && \
18      apt install --yes python3-pip \
19                        libpython3-dev && \
20      pip3 install "cutadapt==$PACKAGE_VERSION" && \
21      apt autoremove --purge --yes && \
22      apt clean && \
23      rm -rf /var/lib/apt/lists/*
24
```

To build cutadapt docker image we will run this command (In this case ciibioinformatics/cuadapt will be the name we have typed in the Docker Hub and we will tag the version number now).
`docker build -t ciibioinformatics/cutadapt:3.0 .` #tag cutadapt version 3.0. The dot is necessary at the end.

#or if dockerfile has another name:

`docker build -t ciibioinformatics/cutadapt:3.0 -f Dockerfile_with_other_name`

Then in order to push it to the Docker Hub:
`docker push ciibioinformatics/cutadapt:3.0`

Instead of creating an image we can also access other images created by other users.
`docker image pull genomicsdocker/fastqc:0.11.5`

Then if we want to push it to Docker hub. We create a new repository in the Hub (in this case fastqc) and run the following commands:

docker tag genomicsdocker/fastqc:0.11.5 ciibioinformatics/fastqc:0.11.5

docker push ciibioinformatics/fastqc:0.11.5

If you want to check all the images available in you docker dashboard:

`docker images`

In order to save the images in tar mode:

`docker save a0260dd94a07 –o prinseq_0.20.3.tar`
; where the number is Image ID that you can check typing docker images.

These images are saved in MAC computer here:
~/Library/Containers/com.docker.docker/Data/vms/0

Depending on which system operator you have, the image tar files will be saved in different locations. Check this link:
https://www.freecodecamp.org/news/where-are-docker-images-stored-docker-container-paths-explained/

If you would like to run any of the containers through docker. For example NCBI:

docker run --rm ncbi/blast update_blastdb.pl --showall

Documentation for how to use docker is in:

https://docs.docker.com/get-started/overview/
https://www.melbournebioinformatics.org.au/tutorials/tutorials/docker/media/#51

Since it is not safe to use Docker in the server we will use singularity.
Here are some links to check on how to work with singularity:
https://researchcomputing.princeton.edu/support/knowledge-base/singularity

We log into the server:
ssh maider@ciilogin.c2b2.columbia.edu

And we go to softwares/singularity to create the singularity images. Singularity is located in /usr/bin/singularity.

cd /share/data/softwares/singularity

To create a singularity image from Docker hub we do it as follows in the current location.
singularity pull docker://ciibioinformatics/bowtie:v2.4.1_cv1

However, the first time we got an error since in the temporary folder there wasn't enough space.

```
[maider@login singularity_images]$ singularity pull docker://ciibioinformatics/bowtie:v2.4.1_cv1
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob 9ff7e2e5f967 done
Copying blob 59856638ac9f done
Copying blob 6f317d6d954b done
Copying blob a9dde5e2a643 done
Copying blob 675cac559d07 done
Copying blob 0e6dd20ee818 done
Copying blob 374c558e71da done
Copying blob 0df3c64bb19a done
Copying blob e936d7784cf9 done
Copying blob 4dfd8d164cc5 done
Copying blob 473490461998 done
Copying blob 8f5e491552e6 done
Copying blob a66ab3a674d9 done
Copying blob 18f922275a6f done
Copying blob be2c066791ee done
Copying blob 2c4fce6ab6cb done
Copying blob 3c22c5023d73 done
Copying blob c77dd2964732 done
Copying config 32a489bded done
Writing manifest to image destination
Storing signatures
FATAL:   While making image from oci registry: error fetching image to cache: while building SIF from layers: conveyor failed to get: Error w
riting blob: write /tmp/bundle-temp-335623689/oci-put-blob393417325: no space left on device
```

So what I have done is to change the temporary location in my bash profile:

```
#change temporary folder location to have more space.
export TMPDIR=/share/data/tmp
```

And then run in qlogin mode:
qlogin -l mem=30G
singularity pull docker://ciibioinformatics/bowtie:v2.4.1_cv1

Singularity objects also can be created in own computer and then upload to server:
`singularity build [IMAGE NAME].sif docker-daemon://[IMAGE NAME]:latest`

Running nextflow in the server

Nextflow will be run in /share/data/ folder.

The main command will be lunch using a sh script. Nextflow takes the configuration "nextflow.config" file as default which has to be in the same folder.

The Shell script to summit the job will be nextflow.sh

```
#!/bin/bash
#
#$
#$ -cwd
#$ -j y
#$ -N illumina_nextflow
#$ -S /bin/bash
#$ -l h_vmem=80G

## backup dir format ##
temp_dir=$(date '+%d_%m_%y_')$RANDOM

module load /usr/bin/singularity
/share/data/software/nextflow/nextflow run Nextflow_v6.nf -with-trace -w temp_${temp_dir}

#done
nextflow.sh (END)
```

In this case the main script is ("Nextflow.nf") and config file is ("nextflow.config"). Nextflow will create a work directory where all the steps will be printed. We can follow the identification of each step in the log that creates the pipeline.

Nextflow config is as follows (version in development):

```
manifest {
  description = 'RNAseq analysis with Nextflow'
  mainScript = 'Nextflow_v3.nf'
}

docker {
  enabled= false
}

singularity {
  enabled=true
  runOptions = '-B /share/data'
}

//configuration for each process
process {
    withName:concatanate_fastq {
       process.executor = 'sge'
       clusterOptions = '-l h_vmem=10G'
    }
    withName:adaptor_trimming {
       container = '/share/data/software/singularity/cutadapt_3.0.sif'
       process.executor = 'sge'
       clusterOptions = '-l h_vmem=10G -pe smp 2'
    }

    withName:count_cutadapt {
       process.executor = 'sge'
       clusterOptions = '-l h_vmem=10G -pe smp 2'
    }
```

In this file we will contain the parameters for input locations, output locations or specific processes parameters such as the parameters for prinseq command.

In this case as input we will need a genome fasta file in “data” directory, fastq files in “Raw_fastq” and adator_3prim.fa file in the working directory.

- In the data folder we will have one virus genome. In my case I copied the influenza genome from /share/data/custom_db_blast/FLU/influenza.fna.
- In Raw_fastq I have war fastq files from one of the last projects: /share/data/IlluminaData/BRD/NextSeq_May11_2022/Raw_fastq
- adaptor_Illumina.list which contain in fasta format adaptor sequences used in the studies

The second file is the main script in which we will place the main steps of the script.

Users > MAIDER > Documents > nextflow > fastq_Bowtie2 > CLIA_Nextflow_v1.nf

```
/*
*   CLIA pipeline nextflow
*/


log.info """\
         CLIA-NF   P I P E L I N E
         ================================
         genome: ${params.genome}
         fastq.reads: ${params.raw}
         output.quality.report: ${params.quality}
         output.trimmed.reads: ${params.trimmed}
         output.index.directory: ${params.index}
         output.alignment.directory: ${params.alignment}

         """
         .stripIndent()

//parameters are saved in the config file.
genome_file=file(params.genome)
adapter3_file=file(params.adapter3)
result = genome_file.isEmpty()
println result ? "Cannot find reference genome:${params.genome}" : ""

Channel
    .fromPath(params.raw)
    .ifEmpty { error "Cannot find any reads matching: ${params.raw}" }
    .set {fastq_files}

```

The log info will print in the terminal. Then we read the needed parameters from the config file.

```
process adaptor_trimming {
    container 'ciibioinformatics/cutadapt:3.0'
    publishDir params.trimmed, mode:'copy'

    input:
    file reads from fastq_files
    file adapter3 from adapter3_file

    output:
    path "${fastq_id}_trimmed.fastq.gz*" into trimmed_files, trimmed_files2

    script:
    fastq_id=reads.simpleName
    """
    cutadapt -a file:${params.adapter3} -o ${fastq_id}_trimmed.fastq.gz ${reads}
    """


}

process runFastQC {

    container 'genomicsdocker/fastqc:0.11.5'
    publishDir params.quality, mode:'copy'

    input:
    file trim_reads2 from trimmed_files2

    output:
    file("${trim_id2}_fastqc/*.zip") into fastqc_files

    script:
    trim_id2=trim_reads2.simpleName
    """
    mkdir ${trim_id2}_fastqc
    fastqc --outdir ${trim_id2}_fastqc \
    -t 2 \
    ${trim_reads2}
    """
}

process runMultiQC {
    container 'ewels/multiqc'
    publishDir params.quality, mode:'copy'

    input:
    file('*') from fastqc_files.collect()

    output:
    file('multiqc_report.html')

    """
    multiqc .
    """
}
```

Once the fastq files are read the first step will be to remove the adaptors. We will use cutadapt. In this case the cutadapt image was created by us (ciibioinformatics/cutadapt:3.0).

Cutadapt documentation can be found here:
https://cutadapt.readthedocs.io/en/stable/

Then we will run a QC analysis for each of the samples using fastqc and multiqc. In these cases, the images were created from other users and we just pull them.

Output of these processes will be found in results folder.

Next, we will cut the reads by quality and length using prinseq. As well the needed genome will be indexed in order to map the reads.

```
process prinseq_trimming{

    container 'ciibioinformatics/prinseq:0.20.3'

    input:
    file trim_reads from trimmed_files
    publishDir params.trimmed, mode:'copy'

    output:
    file "${trim_id}_good.fastq.gz*" into prinseq_trimming


    script:
    trim_id=trim_reads.simpleName
    """
    zcat ${trim_reads} > unzip
    prinseq-lite.pl -verbose -fastq unzip -graph_data ${trim_id}.gd -out_good ${trim_id}_good -out_bad null -rm_header -no_qual_header -min_len ${params.min_len} -min_qual_mean ${params.min_qual_mean} -l
    gzip ${trim_id}_good.fastq
    """
}

process index {
    container 'biocontainers/bowtie2:v2.4.1_cv1'
    publishDir params.index, mode:'copy'

    input:
    file genome from genome_file

    output:
    path "${genome_name}.*" into index_ch

    script:
    genome_name = genome.baseName
    """
    bowtie2-build --threads 4 -f ${genome} ${genome_name}
    """

}
```

Finally, the trimmed reads will be map to the reference genome and then sam files will be converted into bam files for further used.

```
process alignment {
    container 'biocontainers/bowtie2:v2.4.1_cv1'

    input:
    file prinseq_reads from prinseq_trimming
    file index from index_ch
    file genome from genome_file

    output:
    file "${prinseq_id}_aligned.sam*" into alignment_out

    script:
    genome_name = genome.baseName
    prinseq_id=prinseq_reads.simpleName
    """
    bowtie2 -x ${genome_name} -U ${prinseq_reads} -S ${prinseq_id}_aligned.sam
    """

}

process samtobam {
    container 'jweinstk/samtools:latest'
    publishDir params.alignment, mode:'copy'

    input:
    file sam_reads from alignment_out

    output:
    file "${sam_id}.bam*" into alignment_out2


    script:
    sam_id=sam_reads.baseName
    """
    samtools view -S -b $sam_reads > ${sam_id}.bam
    """

}
```