Hackathon reproductible, AMI2B 2020

Release 1.0

Frederic Lemoine, Thomas Cokelaer

Oct 02, 2020

CONTENTS

CHAPTER

ONE

BEFORE WE START

To follow this module, you will need the following prerequisites:

1. An active account on the IFB cloud (BioSphere);
2. A github account;
3. A Integrated Development Environment (IDE).

The whole procedure to run a virtual machine on the IFB cloud is described in *IFB Cloud*.

If you do not have a GitHub account, you can create one here . Then send us your Github username (frederic.lemoine@pasteur.fr, thomas.cokelaer@pasteur.fr), we will add you in the organization.

We will use Visual Studio Code (vscode) as an IDE. It supports large amount of programming languages, version control (git), etc. You can download and install vscode for Linux, MacOS and Windows here.

CHAPTER

TWO

IFB CLOUD

2.1 What is the IFB Cloud?

Informations taken from documentation .

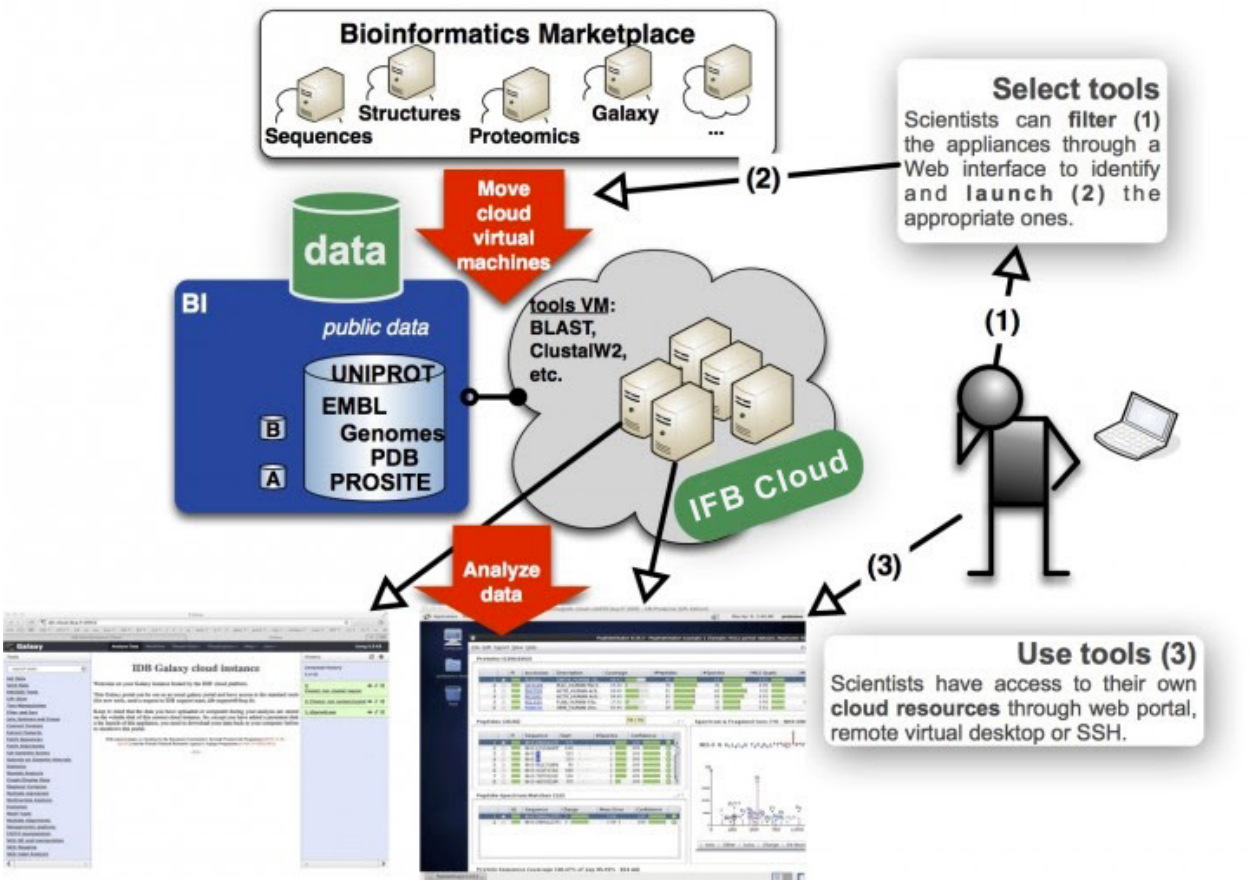
"French Institute of Bioinformatics (IFB) provides life scientists with a federation of clouds, Biosphere, and bioinformatics cloud services to analyze life science data. Biosphere is used for scientific production in the life sciences, developments, and to support events like cloud and scientific training sessions, hackathons or workshops."

"Le cloud IFB comprend 6 000 cœurs de calcul et 28 teraoctets (To) de mémoire. Ces ressources sont réparties entre 7 sites, le cloud de l'IFB-core et 6 Clouds de plateformes régionales de l'IFB et de mésocentres de calcul (GenOuest, PRABI-LBBE, BiRD, BiGest, Bilille, CBP-PSMN), dont 2 en lien avec le GIS France Grilles."

"Appliances" (Virtual machine images) are listed on an online catalogue: RainBio .

Table 1: IFB Cloud federation

Plateforme	Localisation	Calcul (#CPU HT*)	Stockage (#TB)	RAM (#GB)
Fédération de clouds		6080	1026	28540
IFB Core	Lyon (CC-IN2P3)	3936	408	20408
GenOuest	Rennes	600	350	2600
PRABI	Lyon	448	144	1500
BiRD	Nantes	512	50	2048
BIGEst	Strasbourg	200	50	1024
BILILLE	Lille	192	0	768
CBP-PSMN	Lyon	192	24	192
Total (clusters + clouds)		21 942	11 272	115 964



2.2 Install/Configure an SSH client

2.2.1 Windows

Install MobaXterm

MobaXterm is a portable SSH client for windows that will allow you to connect to the IFB cloud.

Link to download MobaXterm

Create a new key pair

To create new encryption keys:

1. Go to `Tools/MobaKeyGen`;
2. Click `Generate`;
3. Click `Save public key` and `Save private key`.
4. Copy the string version of the public key (top of the window), it starts with `ssh-rsa ...`

2.2.2 Linux / MacOSX

SSH client

Under Linux and MacOSX, a client is already installed. You can access it via the Terminal,and the `ssh` command.

Create a new key pair

If not already done (see if a file named `id_rsa.pub` already exists in `$HOME/.ssh/`), you can generate a new SSH public/private key pair with:

```
$ ssh-keygen -t rsa
```

And follow the instructions. You can then copy the string version of your new public key by copying the content of the file `$HOME/.ssh/id_rsa.pub`.

2.3 Connect to *Biosphere*

2.3.1 Login to *Biosphere*

Log into *Biosphere* using your academic account (University, Eduroam, etc.).

Then click on the *Groupes* parameter in your account, and ask to join the group `AMI2B_Hackathon`. Your application will be pending until I accept it.

2.3.2 Add SSH Public Key to your account

Go to your account `parameters`, click edit, and paste the public key previously copied (*ssh-rsa ...*).

2.4 Launch and connect a virtual machine

2.4.1 Launch the VM

Go to the RainBio tab on *Biosphere*, and select the VM (appliance) named *BioPipes*. Then launch the appliance while carefuly selecting:

1. Groupe à utiliser: AMI2B;
2. Cloud: ifb-core-cloud;
3. Gabarit d'image cloud: "ifb.m4.4xlarge".

Once the appliance launched (it can take some time, 20 minutes is not impossible), you can copy the machine IP address : `xxx.xxx.xxx.xxx`.

2.4.2 Connect to the VM

Windows

Go back to MobaXterm, click on the `Session` icon, and `SSH`. Fill the following fields:

1. Remote host: IP address copied from the running appliance page;
2. Click `Specify User Name`, and write `ubuntu`;
3. Click on `Advanced SSH settings` , on `Use Private Key`, and upload the private key file previously exported;
4. Click OK, and the SSH session on the VM should open.

Linux/MacOSX

1. Open a Terminal
2. Run ssh: `ssh -Y ubuntu@xxx.xxx.xxx.xxx`

The right private key should be automaticaly imported.

CHAPTER

THREE

DOCKER

3.1 Example of a Dockerfile

```
# base image: Ubuntu
FROM ubuntu:16.04
RUN apt-get update --fix-missing \
&& apt-get install -y wget gcc make libbz2-dev zlib1g zlib1g-dev liblzma5 liblzma-dev␣
↪libncurses5 libncurses5-dev bzip2 \
&& cd /usr/local/ \
&& wget -O samtools.tar.bz2 https://github.com/samtools/samtools/releases/download/1.
↪9/samtools-1.9.tar.bz2 \
&& tar -xjvf samtools.tar.bz2 \
&& rm -rf samtools.tar.bz2 \
&& cd samtools-1.9 \
&& ./configure \
&& make \
&& make install \
&& cd /usr/local \
&& rm -rf /usr/local/samtools-1.9 \
&& apt-get remove -y wget gcc make libbz2-dev zlib1g-dev liblzma-dev libncurses-dev␣
↪bzip2 \
&& apt-get autoremove -y \
&& apt-get clean \
&& rm -rf /var/lib/apt/lists/* \
&& mkdir /pasteur
ENTRYPOINT ["/usr/local/bin/samtools"]
```

This builds and installs samtools on a blank ubuntu. Here, all commands are executed on only one `RUN` instruction, but each command may be run in a separate `RUN` instruction. In that case, changing one line of the Dockerfile will affect only following instructions at build stage.

3.2 Building and managing images

We can build the image with:

```
docker build . -t evolbioinfo/samtools:v1.9
```

This will create an image called `samtools` version `v1.9` linked to the account `evolbioinfo`, and will execute all the instructions listed in the Dockerfile.

We can also pull already built image from public repositories (such as DockerHub):

```
docker pull hello-world
```

We can list locally installed Docker images with:

```
docker images
```

We can also push locally created images to DockerHub with:

```
docker login
docker push evolbioinfo/samtools:v1.9
```

3.3 Running containers

To run a command inside a container:

```
docker run -v $PWD : $PWD -- entrypoint bash -i -t evolbioinfo/samtools:v1.9
```

It will:

1. Run a samtools container
2. Execute bash

Command options:

- `-v`: bind local folders to container folders
- `-i`: Interactive (Keep STDIN open)
- `-t`: Allocates a pseudo TTY
- `--entrypoint bash`: Changes the default entrypoint
- With default entrypoint, just like calling samtools locally
- `-p`: Redirects local ports to container ports (useful for web services)

3.4 Removing containers/images

We list containers, and then remove them by ID:

```
docker container ls -a
docker rm -vf <ID>
```

Then list and remove images:

```
docker images
docker rmi <ID>
```

3.5 Practical

The goal is to create a Docker image containing kallisto, a bioinformatic tool that quantifies abundances of transcripts from RNA-Seq data.

1. Start writing a Dockerfile starting from our `samtools` example
2. Write RUN commands to get kallisto executable and put it on the PATH (`/usr/local/bin` for example)
3. Build the image
4. Run kallisto help command via Docker : `-h`
5. Remove the kallisto container and the kallisto image

CHAPTER

FOUR

GIT TUTORIAL/EXERCICES

4.1 prerequisite

Connect to IFB Cloud

- Under windows: MobaXterm
- Under linux: `ssh -Y ubuntu@xxx.xxx.xxx.xxx`

Then install an editor on the VM:

```
$ sudo apt-get install mousepad
$ mousepad
```

4.2 cloning an existing repository

Git is a standalone tool. Don't be confuse with github/gitlab. Github is an online website to store git repositories.

Here, we can start by cloning an existing public repository locally

```
git clone https://github.com/SandboxRepro2020/sandbox
```

You can now go to the directory:

```
# under linux:
cd sandbox
```

An play with the content of this repository. There is nothin special here. This is a public repository but you can not push since you do not have the right. Howeve, you can add/commit files locally.

If you had permissios you could then push on the repository.

4.3 Work alone on your private repository

You can create a repository in your github account. If not done already, create an account on github. Then, in your account, create a repository called **sandbox**:

```
git clone https://github.com/your_login/sandbox
```

At that stage, you will be asked to enter your login and password. Once you have cloned the repository, go inside:

```
# under linux, type
cd sandbox
```

This is empty of course. Time to add a new file. A good practice is to add a README file. You may add an extension to indicate the syntax of your file. It may be *.rst* for restructuredText or *.md* for markdown. Here we use *.rst*.

Note: to know more about restructured text, have a look at this Tutorial . A bit old but should be a good starting point.

Create a file called README.rst and add some text inside. Once ready, you can add it into your local git repository. The first time you must **add** it:

```
git add README.rst
```

To validate the change, you will have to **commit** your modification:

```
git commit README.rst
```

Finally, to make sure it is safe, push it online in your github repository:

```
git push
```

Check on your https://github.com/login/sandbox that the README is there. You may edit the README online. If so, at some point you may want to synchronize your local repository:

```
git pull
```

If changes have been made on the server, you should get the changes. Otherwise, well, nothing will happen.

Once synchronised, let us change the README.rst locally. Add some new text/info. Check that there are changes with the status command:

```
git status
```

you should see that the file has changed indeed. You can commit/push again on the server:

```
git commit README.rst
git push
```

Note that this time, there was no need to use *git add* since the README.rst was already added earlier.

4.4 Work together

Invite someone from the group to your repository:

https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/inviting-collaborators-to-a-personal-repository

- Your new collaborator can now clone the repository. Locally She/He can add a new file and commit/psuh the file to your repository.
- You may now retrieve the file and change the file as well.

4.5 Recipes: connect a local repository folder to your empty folder/repository on Github.

Sometimes, you have a nice package on your laptop that you have started from scratch. You suddely realised that you have done lots of work and would like to push it on your github account.

It is possible of course. You can start a local project and later upload it online.

Imagine this local repository called *repro*. You can initialise it and add a new file as follows:

```
mkdir repro
cd repro
git init
git add README
git commit README
git push
```

This is a local repository. nobody knows about it. Better to push it on external server.

First, you need to create a repository on github. Let us call it **repro** as well.

Copy the link in the input right beneath the title, it should look something like this: https://github.com/yourlgin/repro.git

This is the github web address where your local folder can be pushed to and linked to at the same time (as if you did a clone).

Go back to your project in the terminal/command line and type this esoteric command:

```
git remote add origin https://github.com/yourlogin/repo.git
```

Push your branch to Github:

```
git push origin master
```

Go back to the folder/repository screen on Github that you just left, and refresh it. You should see the files in the web interface.

CHAPTER

FIVE

SINGULARITY

Please see the slides about singularity before starting the following exercices.

5.1 Prerequisites

Install singularity under a Linux machine.

On the cloud:

```
$ ## Activate the conda environment
$ conda activate
$ ## Install singularity (just once per newly activated VM)
$ conda install singularity=3.6.3
$ ## Display singularity help
$ singularity --help
```

5.2 Create fastqc container

First, following examples from the course, we should build a container with the fastqc executable in it.

Here is a recipe to install **fastqc** under a centos Linux box:

```
Bootstrap: docker
From: centos:7

%post
yum install -y java-1.8.0-openjdk
yum install -y wget unzip perl
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.9.zip
unzip fastqc_v0.11.9.zip
chmod 755 /FastQC/fastqc

%environment
    export PATH=/FastQC:$PATH

%runscript
    exec fastqc "$@"
```

Once you have that file, name it Singularity.fastqc. You can then create the container with this command:

```
sudo singularity build fastqc.img Singularity.fastqc
```

Try the container:

```
singularity run fastqc.img --help
```

5.3 Create STAR container

It's your turn. Using the same recipes as above and resources from internet, figure out how to install STAR starting from https://github.com/alexdobin/STAR

5.4 Create feature_counts container

Same exercice for feature_counts, a tool from the subread package: http://subread.sourceforge.net/

CHAPTER

SIX

NEXTFLOW

6.1 Example of a Nextflow workflow

main.nf:

```
fastaFiles = Channel.fromPath("/home/user/fasta/*.fasta")

process reformatFasta {
  input:
  file sequences from fastaFiles

  output:
  file "*.phylip" into phylipFiles

  script:
  """
  goalign reformat phylip -i ${sequences} -o ${sequences.baseName}.phylip
  """
}
```

nextflow.config:

```
executor {
    name = 'slurm'
    queueSize = 2000
}
docker {
    enabled = true
}
report {
    enabled = true
    file = 'reports/report.html'
}

trace {
    enabled = true
    file = 'reports/trace.txt'
}

timeline {
    enabled = true
    file = 'reports/timeline.html'
}
```

(continues on next page)

(continued from previous page)

```
dag {
    enabled = true
    file = 'reports/dag.dot'
}

process {
    executor='local'
    scratch=false
    maxRetries=30
    errorStrategy='retry'

    withName: reformatFasta {
        cpus=1
        memory "1G"
        time "30s"
        container="evolbioinfo/goalign:v0.3.2"
    }
}
```

6.2 Run the workflow

`nextflow run main.nf`

Nextflow options:

- `-resume` : restarts an analysis without reexecuting tasks there were succesful or that are not impacted by the workflow update
- `-w <dir>` : Changes the default work directory (default: `work/`)
- `-C <file>`: Changes the default configuration file (default: `nextflow.config`)

6.3 Documentation

https://www.nextflow.io/docs/latest/

CHAPTER

SEVEN

SNAKEMAKE

In the following, we will play with two small FastQ files. Let us download them:

1. https://tinyurl.com/y4uaydth
2. https://tinyurl.com/yxcj6q36

Download the two fastq files and unzip them for the next exercice.

7.1 Exercice: execution

Adapt the following following workflow to analyse the two files you have just downloaded.

```
samples = ['A', 'B']

rule all:
  input:
    expand("stats/{sample}.txt", sample=samples)

rule count:
  input: "{sample}.fastq"
  output: "stats/{sample}.txt"
  run:
    count = 0
    with open(input[0], 'r') as fin:
        for line in fin.readlines():
            count += 1
    with open(output[0], "w") as fout:
        N = int(count / 4)
        fout.write("{}".format(N))
```

Then execute the pipeline and get a flavor of what is going on.

```
snakemake -s name.workflow --cores 1
```

7.2 Exercice (deal with error)

Copy and try this Snakemake workflow. First get the two input files A.fastq.gz and B.fastq.gz in the local directory:

```
samples = ['A', 'B']

rule all:
    input:
        expand("fastqc/{sample}_fastqc.html", sample=samples)

rule fastqc:
    input: "{sample}.fastq.gz"
    output:
        html="fastqc/{sample}_fastqc.html",
        zip="fastqc/{sample}_fastqc.zip"
    log: "logs/{sample}.log"
    shell: "fastqc {input} >{log} 2>&1"
```

what is wrong with the workflow ? Look at fastqc documentation and change the pipeline to fix it. There are two solutions. One that consists in changing the two rules input/output. The second solution need to change only one line by adding 9 characters (only).

7.3 Exercice (benchmark)

From the fastqc example here below:

```
samples = ['A', 'B']
rule all:
    input:
        expand("fastqc/{sample}_fastqc.html", sample=samples)

rule fastqc:
    input: "{sample}.fastq.gz"
    output:
        html="fastqc/{sample}_fastqc.html",
        zip="fastqc/{sample}_fastqc.zip"
    threads: 4
    log: "logs/{sample}.log"
    shell: """fastqc {input} -o fastqc -t {threads} >{log} 2>&1 """
```

add a benchmark (see the course material). Repeat the experiment 5 times.

Execute the pipeline using 4 cores then 8 cores. Check from the benchmark output files that indeed you have 5 entries indicating the time that each run took.

CHAPTER

EIGHT

PROJECT

8.1 Workflow to implement and execute

8.2 Downloading FASTQ files

- Container: `evolbioinfo/sratoolkit:v2.5.7`
- Commands:

```
$ SRAID=SRR....
    $ PREFIX1=${SRAID:0:3}
    $ PREFIX2=${SRAID:0:6}
    $ wget anonymous@ftp.ncbi.nlm.nih.gov:/sra/sra-instant/reads/ByRun/sra/$PREFIX1/
↪$PREFIX2/$SRAID/$SRAID.sra
    $ fastq-dump --gzip --split-files ./$SRAID.sra
```

8.3 Downloading chromosome sequences

- Container: `evolbioinfo/ubuntu:v2.5.7`
- Chromosome names: "chr1", "chr2", "chr3", "chr4", "chr5", "chr6", "chr7", "chr8", "chr9", "chr10", "chr11", "chr12", "chr13", "chr14", "chr15", "chr16", "chr17", "chr18", "chr19", "chr20", "chr21", "chr22", "chrM", "chrX", "chrY"
- Commands:

```
$ wget http://hgdownload.soe.ucsc.edu/goldenPath/hg19/chromosomes/<chromosome>.fa.gz
$ gunzip -c *.fa.gz > ref.fa
```

8.4 Creating genome index

- Container: `evolbioinfo/star:v2.7.6a`
- Commands:

```
$   STAR --runThreadN <nb cpus> --runMode genomeGenerate --genomeDir ref/ --
↪genomeFastaFiles ref.fa
```

8.5 Getting genome annotations (GFF)

- Container: `evolbioinfo/ubuntu:v2.5.7`

8.6 Mapping FastQ files

- Container 1: `evolbioinfo/star:v2.7.6a`
- Container 2: `evolbioinfo/samtools:v1.11`
- Commands:

```
$ STAR --outSAMstrandField intronMotif \
    --outFilterMismatchNmax 4 \
    --outFilterMultimapNmax 10 \
    --genomeDir ref \
    --readFilesIn <(gunzip -c <fastq1>) <(gunzip -c <fastq2>) \
    --runThreadN <Nb CPUS>  \
    --outSAMunmapped None \
    --outSAMtype BAM SortedByCoordinate \
    --outStd BAM_SortedByCoordinate \
    --genomeLoad NoSharedMemory \
    --limitBAMsortRAM <Memory in Bytes> \
    > <sample id>.bam
$ samtools index *.bam
```

8.7 Counting reads

- Container: `evolbioinfo/subread:v2.0.1`

```
$ featureCounts -T 4 -t gene -g ID -s STRAND\
    -a input.gff -o output.counts input.bam""")
```

1. **-t gene** indicates that counts should be done on gene. You may use other features such as **exon**
2. **-g ID** selects the gene ID to store in the output file. This depends on your input GFF file .
3. **STRAND** should be set to 0, 1, 2 depending on strandness of the data.

8.8 Statistical analysis (differential gene expression)

- Container: `evolbioinfo/deseq2:1.28.1`