TD assemblage NGS : découverte de l'environnement en ligne de commande **conda**

Création d'un environnement de travail « bacterial_genome »

1) Création d'une VM ubuntu (sous Windows10) :
2) Création d'un fichier d'environnement (python = 3.6…)
3) Installation de miniconda
4) Installation d'outils de contrôle-qualité (ex : quast) dans l'environnement de travail (base):

Installation de quast à partir des sources:

*(base)$ pip install quast*

*Building wheels for collected packages: quast*

*Building wheel for quast (setup.py) ... error*

*ERROR: Command errored out with exit status 1: RuntimeError: This setup.py does not support wheels. setup.py install will be run automatically...*

*Installing collected packages: quast*

*Running setup.py install for quast ... done*

*DEPRECATION: quast was installed using the legacy 'setup.py install' method, because a wheel could not be built for it. pip 21.0 will remove support for this functionality. A possible replacement is to fix the wheel build issue reported above. You can find discussion regarding this at https://github.com/pypa/pip/issues/8368.*

*Successfully installed quast-5.0.2*

5) Création d'un fichier test de définition d'environnement ($ nano environment.yml)
   *name: bacterial_genome*
   *channels:*
   *- defaults*
   *dependencies:*
   *- python=3.6*
   *- pip*
   *- matplotlib-base*
   *- joblib*
   *- simplejson*
   *- zlib*
   *- perl*
   *- openjdk >= 8*
   *prefix: /opt/bacterial_genome*

6) Création de l'environnement de travail pour l'assemblage : ***$ conda env create –name bacterial_genome -f environment.yml***
7) Activation de cet environnement : *$ conda activate bacterial_genome*
8) Installation des outils de nettoyage trimmomatic : $ conda install trimmomatic=
9) Export de l'environnement de travail pour qu'il soit réutilisable par une autre personne : $ conda env export –file TD.yml

   Le fichier exporté est présenté sur la page suivante à titre d'exemple, afin de pouvoir recréer un environnement de travail fonctionnel sur une nouvelle machine virtuelle Ubuntu (ex : sur un second PC équipé de Win10) sur lequel on aura au préalable installé conda

```
name: bacterial_genome
channels:
  - bioconda
  - defaults
dependencies:
  - _libgcc_mutex=0.1=main
  - blas=1.0=mkl
  - ca-certificates=2020.12.8=h06a4308_0
  - certifi=2020.12.5=py36h06a4308_0
  - cycler=0.10.0=py36_0
  - fastqc=0.11.9=0
  - font-ttf-dejavu-sans-mono=2.37=h6964260_0
  - fontconfig=2.13.0=h9420a91_0
  - freetype=2.10.4=h5ab3b9f_0
  - icu=58.2=he6710b0_3
  - intel-openmp=2020.2=254
  - joblib=1.0.0=pyhd3eb1b0_0
  - jpeg=9b=h024ee3a_2
  - kiwisolver=1.3.0=py36h2531618_0
  - lcms2=2.11=h396b838_0
  - ld_impl_linux-64=2.33.1=h53a641e_7
  - libedit=3.1.20191231=h14c3975_1
  - libffi=3.3=he6710b0_2
  - libgcc-ng=9.1.0=hdf63c60_0
  - libpng=1.6.37=hbc83047_0
  - libstdcxx-ng=9.1.0=hdf63c60_0
  - libtiff=4.1.0=h2733197_1
  - libuuid=1.0.3=h1bed415_2
  - libxml2=2.9.10=hb55368b_3
  - lz4-c=1.9.2=heb0550a_3
  - matplotlib-base=3.3.2=py36h817c723_0
  - mkl=2020.2=256
  - mkl-service=2.3.0=py36he8ac12f_0
  - mkl_fft=1.2.0=py36h23d657b_0
  - mkl_random=1.1.1=py36h0573a6f_0
  - ncurses=6.2=he6710b0_1
  - numpy=1.19.2=py36h54aff64_0
  - numpy-base=1.19.2=py36hfa32c7d_0
  - olefile=0.46=py36_0
  - openjdk=8.0.152=h7b6447c_3
  - openssl=1.1.1i=h27cfd23_0
  - perl=5.26.2=h14c3975_0
  - pillow=8.1.0=py36he98fc37_0
  - pilon=1.23=2
  - pip=20.3.3=py36h06a4308_0
  - pyparsing=2.4.7=py_0
  - python=3.6.12=hcff3b4d_2
  - python-dateutil=2.8.1=py_0
  - readline=8.0=h7b6447c_0
  - setuptools=51.1.2=py36h06a4308_4
  - simplejson=3.17.2=py36h7b6447c_0
  - six=1.15.0=py36h06a4308_0
  - sqlite=3.33.0=h62c20be_0
  - tk=8.6.10=hbc83047_0
  - tornado=6.1=py36h27cfd23_0
  - wheel=0.36.2=pyhd3eb1b0_0
  - xz=5.2.5=h7b6447c_0
  - zlib=1.2.11=h7b6447c_3
  - zstd=1.4.5=h9ceee32_0
prefix: /home/isabelle/miniconda3/envs/bacterial_genome
```

importation du précédent environnement sur une nouvelle machine ubuntu (conda installé) :

1) on ajoute quast à la fin du fichier TD.yml importé ($ nano TD.yml puis ajout de la dependance - quast)
2) on va créer l'environnement de travail avec la commande suivante :

$ conda env create -n bacterial_genome -f TD_quast.yml

3) on exporte le nouvel environnement de travail lorsque quast est installé

*voici le nouveau fichier d'environnement Yaml :*

```
name: base
channels:
 - bioconda
 - defaults
dependencies:
 - _libgcc_mutex=0.1=main
 - brotlipy=0.7.0=py38h27cfd23_1003
 - ca-certificates=2020.12.8=h06a4308_0
 - certifi=2020.12.5=py38h06a4308_0
 - cffi=1.14.3=py38h261ae71_2
 - chardet=3.0.4=py38h06a4308_1003
 - conda=4.9.2=py38h06a4308_0
 - conda-package-handling=1.7.2=py38h03888b9_0
 - cryptography=3.2.1=py38h3c74f83_1
 - idna=2.10=py_0
 - ld_impl_linux-64=2.33.1=h53a641e_7
 - libedit=3.1.20191231=h14c3975_1
 - libffi=3.3=he6710b0_2
 - libgcc-ng=9.1.0=hdf63c60_0
 - libstdcxx-ng=9.1.0=hdf63c60_0
 - minia=3.2.4=he513fc3_0
 - ncurses=6.2=he6710b0_1
 - openssl=1.1.1i=h27cfd23_0
 - pip=20.2.4=py38h06a4308_0
 - pycosat=0.6.3=py38h7b6447c_1
 - pycparser=2.20=py_2
 - pyopenssl=19.1.0=pyhd3eb1b0_1
 - pysocks=1.7.1=py38h06a4308_0
 - python=3.8.5=h7579374_1
 - readline=8.0=h7b6447c_0
```

- requests=2.24.0=py_0
- ruamel_yaml=0.15.87=py38h7b6447c_1
- setuptools=50.3.1=py38h06a4308_1
- six=1.15.0=py38h06a4308_0
- sqlite=3.33.0=h62c20be_0
- tk=8.6.10=hbc83047_0
- tqdm=4.51.0=pyhd3eb1b0_0
- urllib3=1.25.11=py_0
- wheel=0.35.1=pyhd3eb1b0_0
- xz=5.2.5=h7b6447c_0
- yaml=0.2.5=h7b6447c_0
- zlib=1.2.11=h7b6447c_3

prefix: /home/erwan/miniconda3

## Conclusion :

L'objectif de ce nouveau TD orienté info était donc de créer un fichier Yaml réutilisable et exportable