Using Insyght
“In the cloud” with the IFB appliance
“Bacterial genomics”
-
User guide
(Version 1.7.1, March 2016)

Citation Insyght:
Lacroix T., Loux V., Gendrault A., Hoebeke M., and Gibrat J-F. (2014) Insyght: navigating amongst abundant homologues, syntenies and gene functional annotations in bacteria, it's that symbol! Nucleic Acids Res. 42(21):e162. PMID: 25249626

Content:

1 Getting started with visualizing your data

~~ Remark: Insyght expects a transcript per CDS and is therefore only suitable for Bacteria and Archea.

1.1 Launch a "Bacterial genomics" appliance

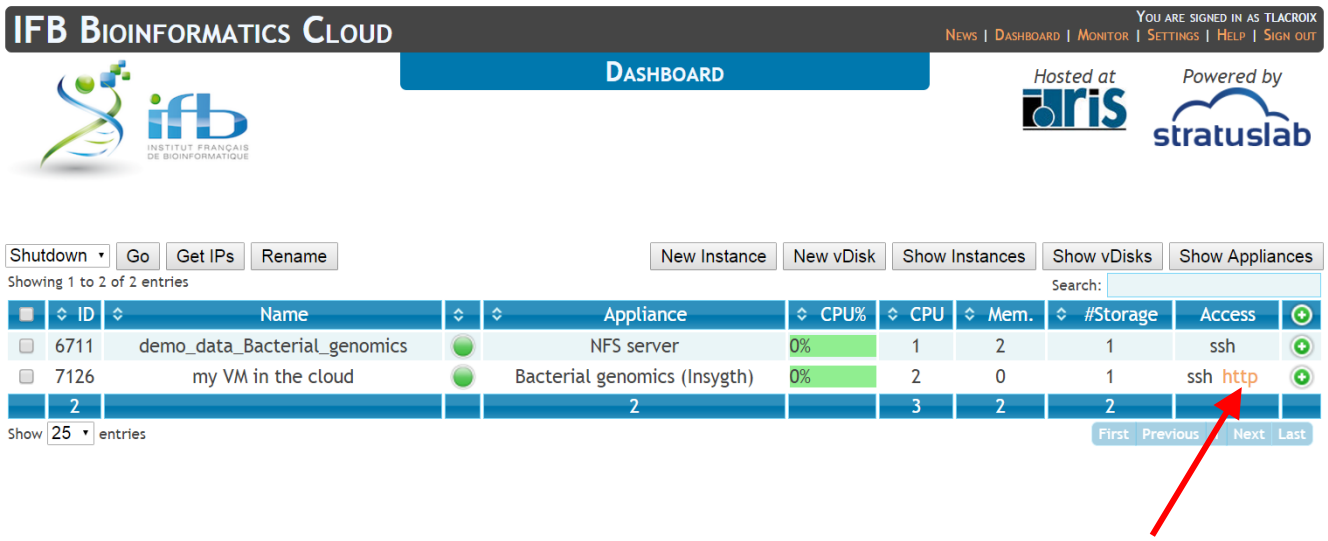
Login to the IFB Bioinformatics Cloud at https://cloud.france-bioinformatique.fr/cloud/. Create a new vDisk from the dashboard (click on "New vDisk") and give it a name and a size. Click on "New Instance", choose the appliance "Bacterial genomics", give it a name, plug in the vDisk you just created (section "Plug Your Additional Storage,-> Persistent disk"), and launch the virtual machine (VM) by clicking "Run". Allow a few minutes for the virtual machine to start.

~~ Remark: It is recommended to create a vDisk of at least 3 Go for a dataset of ~15 genomes. The vDisk space needed grows exponentially with the number of genomes in the dataset. For example a dataset of 350 genomes requires a vDisk of ~3000 Go. The tmp files generated by the pipeline account for ~60% of the disk space. This space will be freed at the end of a pipeline run (see section "Delete temporary files").

If you populated a vDisk with your own dataset in a previous session and want to work on it or add new genomes, plug this particular vDisk instead of creating a new one.

1.2 Launch Insyght

Click the http link in the "Access" tab; you can right-click -> open in a new tab if you want to keep the IFB cloud's dashboard at hand.



You will be redirected to the Insyght web application associated with this VM.

~~ Remark: You may experience an error message if you try to connect to Insyght and the booting process of the VM is not fully completed. In that case, try to connect again in a few minutes.

If you did plug in a vDisk on which you already inserted data, you should be able to browse them. If you did plug a new vDisk, Insyght features by default a demo dataset of 17 bacterial genomes. This demo dataset is meant to showcase the tool's features.

~~ Remark: Examples of genomes in this demo dataset are: Lactobacillus fermentum, Streptococcus thermophilus, Staphylococcus aureus strain COL, Salmonella newport strain SL254, Mycobacterium tuberculosis strain F11, Listeria monocytogenes serovar 1/2a, strain 08-5578, etc. The genbank, embl (.dat), gbff and fasta files used to generate the demo dataset can be found on the VM in the directory /root/documents/example_files. To access them, see the section "How to connect to your VM in the cloud via a terminal".

Check that the web application is working as expected. If you want to familiarize yourself with the tool's features, see the "Manual on Insyght user interface" at https://migale.jouy.inra.fr/redmine/projects/insyght/documents. To get a grasp of the concepts behind the project, see the guide on "Concepts of homology and synteny in Insyght" (same url).

~~ Remark: You can access this instance of Insyght from any computer connected to the internet. The url provided by the "http" link is active as long as the VM is on.

1.3 How to connect to your VM in the cloud via a terminal

From the IFB Cloud's dashboard, click on the "ssh" link in the "Access" tab. A pop up will appear, copy the text under the section "command-line connection". Open a terminal on your host and paste in the command. You may be asked a confirmation to allow the connection by your host environment.

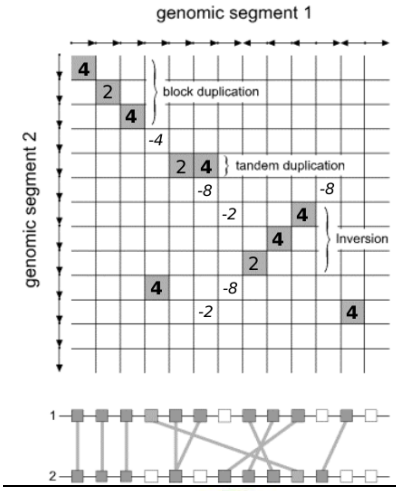
~~ Remark: You may experience a "connection refused" message if you try to connect to the VM and its booting process is not fully completed. In that case, try the command again in a few minutes.

1.4 Overview of the pipeline

The next sections describe in details how to add your own data in Insyght. In short, the database is populated by a pipeline of Perl scripts that:

1. Process the genome files (.embl, .gbk, ou .gbff format) to extract the primary data (genomic annotations).

2. Run the BLASTp jobs and parse the results.



3. Execute the program that identify the syntenies.

4. Insert the data into the relational database.

5. The Insyght web application is used to visualise the data and carry out the data mining.

1.5 How to connect to your instance of the database (PostgreSQL)

In a terminal connected to the VM, type in or paste the following command:
psql -h localhost -p 5432 -U origami_admin -d origami_prod

*~~ Remark: throughout the document, the text that is **highlighted in red** is to be copied/paste or typed in a terminal connected to the VM.*

You are now connected to your instance of the origami database. If you wish, you can use plain sql commands to query the data as you like.

*~~ Remark: type in **\q** to exit the psql mode.*

1.6 [Optional] Remove all current data in your instance of the database

Optionally, you can remove all the current data in your instance of the database before inserting new data. Insyght relies on this database to fetch any information it displays. Skip this section if you want to keep the current data and add yours along with them. The process of adding data in Insyght is incremental and can be done multiple times without deleting the current data.

Connect to your instance of the database and paste the following commands:
SELECT truncate_tables_public('origami_admin');
SELECT truncate_tables_micado('origami_admin');
Exit the psql mode: **\q**

~~ Remark: To get an overview of what is inserted in the database, type in the following command in a terminal:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
The database should now be empty. If you open the Insyght web application, you should see that no organism is available.

1.7 Reclaim disk space of deleted rows and re-calculate the indexes of the database

After some important changes have been carried out to the database, it is best to clean up its underlining storage and indexing mechanisms. Type in the following command in a terminal:
vacuumdb --full --analyze -h localhost -p 5432 -U origami_admin origami_prod

~~ Remark: safely ignore the warnings on the terminal

1.8 Add your own data

This section explains how to add your own data with Insyght.

~~ Remark: we suggest that you try re-loading the demo data first to check that the pipeline is working correctly. See the remark in the steps below to work with the demo data files.

~~ Remark: *if you want to annotate raw fasta files, you can quickly annotate them with prokka which will automatically generate a genbank file. Prokka (http://www.vicbioinformatics.com/software.prokka.shtml) is a software tool for the rapid annotation of prokaryotic genomes developed by the Victorian Bioinformatics Consortium. For more details, see the section "Run prokka to quickly annotate raw fasta file".*

~~ Remark: You can shut down your VM after any step below and continue the process by launching the next step at a later time on a new VM plugged in to the same vDisk (all the necessary information about the state of the process is stored on the vDisk). If an unexpected error occurs during one of the steps described thereafter (i.e. VM crash, etc.), no need to restart the whole process from step 1, just re-launch the step that failed and continue with the process.

~~ Benchmark: Throughout this section, information on benchmarking is provided for each step. The benchmarking was carried out on the following VM: c3.medium (2CPU, 8GB RAM). The number of available internal processors (CPU) in the virtual machine is determined by its type (c2.small = 1 CPU, c3.medium = 2 CPU, etc.).

1.8.1 Delete the temporary and log files generated by the pipeline from the previous run

From a terminal connected to the VM, copy the following commands:
rm -f -r ${DATA_DIR}/*
rm -f -r ${LOG_DIR}/*
rm -f -r ${BANK_DIR}/*

1.8.2 Copy all the .gbk, .dat and .gbff genomes files you want to add

Copy the genomes files (.gbk, .dat, and .gbff) in the directory ${BANK_DIR}.
cp ***PATH_TO_YOUR_FILE* ${BANK_DIR}**
Repeat the command for each file to copy. For example if you did download the annotated genome file for E coli K-12 MG1655 from ncbi (NC_000913.gbk) under the /root/mydisk directory, use the following command:
cp /root/mydisk/NC_000913.gbk ${BANK_DIR}

~~ Remark: The genome files used for the demo data can be found under the directory /root/documents/example_files/genome/
If you want to re-load the demo data, use the command:
cp /root/documents/example_files/genome/* ${BANK_DIR}/

~~ Remark: You can download genome files using a web browser on your host computer or from a terminal connected to the IFB VM via an ftp server. See the section "Different ways to download a raw fasta or annotated genome file from the ncbi or ebi repositories" for more details.

~~ Remark: If you did annotate some fasta file with prokka, do not forget to copy them as well, typically with the command:
cp /root/mydisk/prokka_output/*.gbk ${BANK_DIR}/

~~ Remark: Compressed files with the .gz extension are supported by the pipeline, no need to extract them.

1.8.3 Check and extract the genomes files as necessary

This step will perform preliminary preparations (i.e file extraction etc...) and checking on the genome files to insert.

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
List_check_and_split_genomes_files.pl

~~ Remark: the first command line option is uppercase i, not lowercase l

~~ Remark: the character **** *at the end of a line and inside a command statement is used to split a long command into multiples lines. We use it to improve legibility in this document.*

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see:
List_check_and_split_genomes_files.pl successfully completed at DATE
=> No major problem was detected.
printed on the last line. To access the log file, use
less ${LOG_DIR}/Add_entries/List_check_and_split_genomes_files.log
Warnings will be issued if a specific genome file does not contain sequences or locus tag attributes.

~~ Remark: After this step, gbff files will be expanded into multiple .gbk files (1 file for each accession). Similarly, .multiple_dats files will be expanded into .dat files. You can see the resulting files by typing:
ll ${BANK_DIR}/

~~ Remark: This step generates a file gr2species_migale.txt in the ${BANK_DIR} directory. Lines (<=> accession number) that are deleted or commented by # in this file will be ignored by the subsequent steps of the pipeline.

~~ Benchmark: This step took a few seconds for the files provided as demo (17 organisms total).

~~ Remark: To check the free disk space that remains before or after launching any step, type in the command:
df -h | grep -E '^[F/]'

1.8.4 Insert primary data

This task will insert primary data (organisms, genes, features...) into the database.

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
Task_add_entry.pl

~~ Remark: If you want to restrict the processing of the pipeline to a specific file format, you can add the option **genbank** (*for .gbk files) or* **embl** (*for .dat files) at the end of the command line.*

~~ Remark: The script should start printing information on the terminal as it adds the genomes. When it is finished, it should print:
Task Task_add_entry.pl completed
All entries added at DATE
The script will fail if a genome file is not correctly formatted (missing DNA sequence at the end of the genome file, etc...). The log file for this step is located here:
less ${LOG_DIR}/Add_entries/Add_entries.log

~~ Remark: once the script has finished, you should see that some data has been inserted in the database. To get an overview of what is inserted in the database, type in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
If you previously deleted all the data in the database, the rows count should be 0 for the tables qualifiers, sequences, features, elements, genes, and organisms before launching this step and greater than 0 afterward.

~~ Benchmark: This step took ~5 min for the files provided as demo (17 organisms total).

1.8.5 Correct elements types and reorganize elements into organisms

1.8.5.1 Correct elements generator

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
check_for_element_types_and_reorganize_elements_into_organisms.pl

~~ Remark: The script should start printing information on the terminal as it runs. When it is finished, it should print:
check_for_element_types_and_reorganize_elements_into_organisms.pl
successfully completed at DATE
The log file for this step is located here:
less ${LOG_DIR}/
check_for_element_types_and_reorganize_elements_into_organisms/
check_for_element_types_and_reorganize_elements_into_organisms.log

~~ Remark: To see the file generated by this step, you can use the following commands:
less ${DATA_DIR}/check_reorganize_elements/check_reorganize_elements.sql

~~ Benchmark: This step took a few seconds for the files provided as demo (17 organisms total).

1.8.5.2 Correct elements integrator

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
integrate_changes_element_types_and_reorganize_elements_into_organisms.pl

~~ Remark: The script should start printing information on the terminal as it runs. When it is finished, it should print:
integrate_changes_element_types_and_reorganize_elements_into_organisms.pl successfully completed at DATE
The log file for this step is located here:
less ${LOG_DIR}/
check_for_element_types_and_reorganize_elements_into_organisms/
integrate_changes_element_types_and_reorganize_elements_into_organisms
.log

~~ Benchmark: This step took a few seconds for the files provided as demo (17 organisms total).

1.8.6 Launch task blast

This step will compute secondary data corresponding to the cross comparison, using blast, of all the genes of the stored genomes. Because we use bi-directional best hit (BBDH) as a clue to infer orthology, this information is computed as well.

1.8.6.1 Task blast generator

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
Task_generate_fasta.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
Task_generate_fasta.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_generate_fasta/Task_generate_fasta.log

~~ Remark: To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_blast_all/tmp/fasta_formatdb/ -type f -name '*' | wc -l
→ expected: number of new organisms with genes + 1.
The number of new organisms with genes can be determined by typing in the following command:

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of new organisms with genes.*

**find ${DATA_DIR}/Task_blast_all/tmp/launch_blast_batch_files/makeblastdb/ **
-type f -name '*.makeblastdb.ll' | wc -l
→ expected = 1

~~ Remark: see the section "Changing the threshold parameters for the programs blast (used to find gene similarity) and align (used to compute the synteny)" if you want to customize the e-value cutoff parameters for this step.

~~ Benchmark: This step took a few seconds for the files provided as demo (17 organisms total).

1.8.6.2 Task makeblastdb

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
Task_blast_all_launch_makeblastdb_IDRIS.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
Task_blast_all_launch_makeblastdb_IDRIS.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_blast_all/Task_blast_all_launch_makeblastdb_IDRISS.log

~~ Remark: This step generates the file .phr, .pin, and .psq for the file orga_all.faa. To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_blast_all/tmp/fasta_formatdb/ -type f -name '*' | wc -l
→ expected: number of new organisms with genes + 4.
The number of new organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of new organisms with genes.*

~~ Benchmark: This step took a few seconds for the 17 organisms provided as demo.

1.8.6.3 Task blast generator

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
Task_blast_all_generator_for_IDRIS.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
Task_blast_all_generator_for_IDRIS.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_blast_all/Task_blast_all_generator_for_IDRIS.log

~~ Remark: To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_blast_all/tmp/
launch_blast_batch_files/blast/ -type f -name '*.blast.ll' | wc -l
→ expected: number of new organisms with genes; The number of new organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of new organisms with genes.*

~~ Benchmark: This step took a few seconds for the 17 organisms provided as demo.

1.8.6.4 Task launch blast

rm -f nohup.out

nohup perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
**Task_blast_all_launch_cluster_blast_IDRIS.pl -EMAIL_NOTIFICATION **
****YOUR_EMAIL_ADDRESS@PROVIDER****

~~ Remark: You won't have access to the terminal command line and information will be printed in the file nohup.out while the script run. You will receive an email to notify you when this step is completed. Make sure you provided a correct email address following the option **-EMAIL_NOTIFICATION** *above. If the script is successful, you should see*
Task_blast_all_launch_cluster_blast_IDRIS.pl successfully completed at DATE
printed on the last line of the nohup.out file. To access the log file, use
less ${LOG_DIR}/Task_blast_all/Task_blast_all_launch_cluster_blast_IDRIS.log

~~ Remark: To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_blast_all/tmp/blast_output/ -type f -name '*.blast' | wc -l
→ expected: number of new organisms with genes.
The number of new organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of new organisms*

with genes.

~~ Benchmark: This step took ~7 hours for the 17 organisms provided as demo. Arguments used: **-MAX_JOBS 1** *(or no* **-MAX_JOBS** *argument) and no use of the cluster. As the pipeline may be used incrementally without deleting previous entries, the time for this step may grow longer and longer; the reason is that each new organism has to be compared to all the current entries. To parallelize the processes and make this step run faster, you can use the option* **-MAX_JOBS** *(see below). The total time for this step will decrease proportionally to the maximum number of jobs launched in parallel. With* **-MAX_JOBS 20***, this step took 45 minutes for the 17 organisms provided as demo.*

~~ Remark on the optional argument -MAX_JOBS and -CMD_CLUSTER:
The option **-MAX_JOBS** *determine the maximum number of jobs to be launched in parallel. This value must be at most the number of available processors. To know the number of internal processors available, type in the command:* ***grep -c processor /proc/cpuinfo****. To greatly increase the number of available processors, a solution is to use a cluster (external processors); you can then use the option* **-CMD_CLUSTER***. See the section "Using the IFB cloud cluster (external processors)" for more details. If the* **-CMD_CLUSTER** *argument is omitted (default), then the command is executed by the bash locally on the internal processors.*

~~ Remark: If you want you can close the terminal during that time-consuming step, it will not stop the execution of the jobs. But shutting down the VM will stop all the jobs and result in an incomplete state. If unfortunately the VM has shut down during that step, you can either re-launch this step entirely (same command as above) or launch it as a "recover from failure" mode. To activate this mode, add the argument **-RECOVER_FROM_FAILURE ON** *to the command above. It will then run only the jobs that have not been completed in the previous session.*

1.8.7 Find synteny

1.8.7.1 Syntenies generator

```
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/\
Task_add_alignment_generator_for_IDRIS.pl
```

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
Task_add_alignments_generator_for_IDRIS.pl successfully completed at DATE
printed on the last line. To access the log file, use

```
less ${LOG_DIR}/Task_add_alignments/\
Task_add_alignment_generator_for_IDRIS.log
```

~~ Remark: To count the number of file generated by this step, you can use the

following commands:
**find ${DATA_DIR}/Task_add_alignments/tmp/cmd_files_cluster_mig/ -type f **
-name '*.ll' | wc -l
*→ expected: ((number of new organisms with genes) * (number of new organisms with genes + 1)) / 2 + (number of new organisms with genes * number of previously inserted organisms with genes).*
The number of previously inserted organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of previously inserted organisms with genes.*

***find* ${DATA_DIR}*/Task_add_alignments/tmp/archive_files/ -type f ***
-name '*.archive' | wc -l
→ expected: Total number of organisms with genes.
The number of organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of total number of organisms with genes:*

~~ Remark on the optional arguments:
To change the default parameters used by the program Align to compute the synteny, see the section "Changing the threshold parameters for the programs blast (used to find gene similarity) and align (used to compute the synteny)".

~~ Benchmark: This step took a few seconds for the 17 organisms provided as demo.

1.8.7.2 Launch syntenies finder

rm -f nohup.out

nohup perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
**Task_add_alignment_launch_jobs.pl -EMAIL_NOTIFICATION **
****YOUR_EMAIL_ADDRESS@PROVIDER****

~~ Remark: You won't have access to the terminal command line and information will be printed in the file nohup.out while the script run. You will receive an email to notify you when this step is completed. Make sure you provided a correct email address following the option **-EMAIL_NOTIFICATION** *above. If the script is successful, you should see*
Task_add_alignments_launch_jobs.pl successfully completed at DATE
printed on the last line of the nohup.out file. To access the log file, use

less ${LOG_DIR}/Task_add_alignments/Task_add_alignment_launch_jobs.log

~~ Remark: To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_add_alignments/tmp/tsv/ -type f -name '*.tsv' | wc -l
*→ expected : (((number of new organisms with genes) * (number of new organisms with genes + 1)) / 2 + (number of new organisms with genes * number of previously inserted organisms with genes)) * 4.*
The number of previously inserted organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of previously inserted organisms with genes.*

~~ Benchmark: This step took ~1 hour for the 17 organisms provided as demo. Arguments used: **-MAX_JOBS 1** *(or no* **-MAX_JOBS** *argument) and no use of the cluster. As the pipeline may be used incrementally without deleting previous entries, the time for this step may grow longer and longer; the reason is that each new organism has to be compared to all the current entries. To make this step run faster, you can use the option* **-MAX_JOBS** *(see below). The total time for this step will decrease proportionally to the maximum number of jobs launched in parallel. With 16 jobs launched in parallel (***-MAX_JOBS 16***), this step took ~5 minutes for the 17 organisms provided as demo.*

~~ Remark on the optional argument -MAX_JOBS:
The option **-MAX_JOBS** *determine the maximum number of jobs to be launched in parallel. This value must be at most the number of available processors. To know the number of internal processors available, type in the command:* ***grep -c processor /proc/cpuinfo****. To greatly increase the number of available processors, a solution is to use a cluster (external processors); you can then use the option* **-CMD_CLUSTER***. See the section "Using the IFB cloud cluster (external processors)" for more details. If the* **-CMD_CLUSTER** *argument is omitted (default), then the command is executed by the bash locally on the internal processors.*

~~ Remark: If you want you can close the terminal during that time-consuming step, it will not stop the execution of the jobs. But shutting down the VM will stop all the jobs and result in an incomplete state. If unfortunately the VM has shut down during that step, you can either re-launch this step entirely (same command as above) or launch it as a "recover from failure" mode. To activate this mode, add the argument **-RECOVER_FROM_FAILURE ON** *to the command above. It will then run only the jobs that have not been completed in the previous session.*

1.8.7.3 Generate syntenies parser

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
generate_Task_add_alignment_parse_tsv_output_Replacing_Forward_Reverse
.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
generate_Task_add_alignment_parse_tsv_output_Replacing_Forward_Reverse successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_add_alignments/
generate_Task_add_alignment_parse_tsv_output_Replacing_Forward_Reverse
.log

~~ Remark: To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_add_alignments/tmp/
**launch_parse_tsv_output_Replacing_Forward_Reverse_files -type f **
-name '*.ll' | wc -l
*→ expected: ((number of new organisms with genes) * (number of new organisms with genes + 1)) / 2 + (number of new organisms with genes * number of previously inserted organisms with genes).*
The number of previously inserted organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of previously inserted organisms with genes.*

~~ Benchmark: This step took a few seconds for the 17 organisms provided as demo.

1.8.7.4 Launch syntenies parser

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
launch_Task_add_alignment_parse_tsv_output_Replacing_Forward_Reverse.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
launch_Task_add_alignment_parse_tsv_output_Replacing_Forward_Reverse successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_add_alignments/
launch_Task_add_alignment_parse_tsv_output_Replacing_Forward_Reverse
.log

~~ Remark: To count the number of file generated by this step, you can use the following commands:
find ${DATA_DIR}/Task_add_alignments/tmp/sql/ -type f -name '*.sql' | wc
*→ expected : (((number of new organisms with genes) * (number of new organisms with genes + 1)) / 2 + (number of new organisms with genes * number of previously inserted organisms with genes)) * 4.*
The number of previously inserted organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of previously inserted organisms with genes.*

~~ Benchmark: This step took < 30 seconds for the 17 organisms provided as demo.

1.8.7.5 Drop indexes before database integration

**psql -h localhost -p 5432 -U origami_admin -d origami_prod -a -f **
${ORIGAMI_WORK_DIR}/ORIGAMI/SQL/drop_idx_before_alignment_insert.sql

~~ Benchmark: This step took a few seconds for the 17 organisms provided as demo.

1.8.7.6 Data integration in the database

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
Task_add_alignment_integrator_for_IDRIS.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
Task_add_alignments_integrator_for_IDRIS.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_add_alignments/
Task_add_alignment_integrator_for_IDRIS.log

~~ Remark: once the script has finished, you should see that some data has been inserted in the database. To get an overview of what is inserted in the database, type in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
If you previously deleted all the data in the database, the rows count for the tables alignment_params, alignments, alignment_pairs, and homologies should be 0 before launching the perl script and different than 0 afterward.

~~ Remark: To count the number of file processed by this step, you can use the

following commands:
find ${DATA_DIR}/
Task_add_alignments/tmp/sql/ -type f -name '*.sql.todo_gzip' | wc
*→ expected : (((number of new organisms with genes) * (number of new organisms with genes + 1)) / 2 + (number of new organisms with genes * number of previously inserted organisms with genes)) * 4.*
The number of previously inserted organisms with genes can be determined by typing in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
*And looking at the number under the section ** Counting number of previously inserted organisms with genes.*

~~ Benchmark: This step took ~30 seconds for the 17 organisms provided as demo.

1.8.7.7 Recreate indexes after database integration

**psql -h localhost -p 5432 -U origami_admin -d origami_prod -a -f **
${ORIGAMI_WORK_DIR}/ORIGAMI/SQL/recreate_idx_after_alignment_insert.sql

~~ Benchmark: This step took a few seconds for the 17 organisms provided as demo.

1.8.8 Pre-compute the sorted lists of compared organisms

1.8.8.1 Generate pre-computed lists

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
generate_sorted_list_comp_elements.pl -DO_ALL

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
generate_sorted_list_comp_elements.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/compute/generate_sorted_list_comp_elements.log

~~ Remark: To see the file generated by this step, you can use the following commands:
less ${DATA_DIR}/Task_add_alignments
/tmp/sorted_list_comp_orga_whole/abundance_homologs.sql
less ${DATA_DIR}/
Task_add_alignments/tmp/sorted_list_comp_orga_whole/alignemnt_score.sql
less ${DATA_DIR}/
Task_add_alignments/tmp/sorted_list_comp_orga_whole/synteny_score.sql

~~ Benchmark: This step took a few seconds for the files provided as demo (17

organisms total).

1.8.8.2 Integrate pre-computed lists

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
integrate_sorted_list_comp_orga_whole.pl -DO_ALL

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
integrate_sorted_list_comp_orga_whole.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/compute/integrate_sorted_list_comp_orga_whole.log

~~ Remark: once the script has finished, you should see that some data has been inserted in the database. To get an overview of what is inserted in the database, type in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
If you previously deleted all the data in the database, the rows count should be 0 for the table q_element_id_2_sorted_list_comp_orga_whole before launching the perl script and different than 0 afterward.

~~ Benchmark: This step took a few seconds for the files provided as demo (17 organisms total).

1.8.9 Create the taxonomic tree

1.8.9.1 Generate the taxonomic tree

The virtual machine needs to be connected to the Internet for this step else the wget command will fail.
mkdir -p ${DATA_DIR}/Update_taxo/
rm -fr ${DATA_DIR}/Update_taxo/*
cd ${DATA_DIR}/Update_taxo/
wget -nv -N ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz
[this command might take a few second to complete]
tar xzf taxdump.tar.gz
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
ParseNCBITaxidFiles.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
ParseNCBITaxidFiles.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Update_taxo/ParseNCBITaxidFiles.log

~~ Remark: To see the file generated by this step, you can use the following commands:
less ${DATA_DIR}/Update_taxo/taxo.dat
less ${DATA_DIR}/Update_taxo/taxo_names.dat

~~ Benchmark: This step took ~2 minutes for the files provided as demo (17 organisms total).

1.8.9.2 Integrate the taxonomic tree

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
integrate_taxo_data.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
integrate_taxo_data.pl successfully completed at DATE
printed on the last line. To access the log file, use
less ${LOG_DIR}/Update_taxo/integrate_taxo_data.log

~~ Remark: once the script has finished, you should see that some data has been inserted in the database. To get an overview of what is inserted in the database, type in the following command:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
count_inserted_data.pl
If you previously deleted all the data in the database, the rows count should be 0 for the table taxo and taxo_names before launching the perl script and different than 0 afterward.

~~ Benchmark: This step took a few seconds for the files provided as demo (17 organisms total).

1.8.10 Reclaim database disk space and update indexes

vacuumdb --full --analyze -h localhost -p 5432 -U origami_admin origami_prod

~~ Remark: safely ignore the warnings on the terminal

~~ Benchmark: This step took ~1 minute for the 17 organisms provided as demo.

1.8.11 Check the consistency of the database

This step will check for redundancies, missing data, and errors that may have occurred during the insertion of the data in the database. If this script fails then the application is unstable.

perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
check_db_consistency.pl

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see
check_db_consistency.pl successfully completed at DATE
Summary :
No major problem was detected, your database is stable.
No warning was detected.
printed on the last line. To access the log file, use
less ${LOG_DIR}/Task_add_alignments/check_db_consistency.log

~~ Benchmark: This step took <10 seconds for the 17 organisms provided as demo.

1.8.12 Delete temporary files

rm -f -r ${DATA_DIR}/*
rm -f -r ${LOG_DIR}/*
rm -f -r ${BANK_DIR}/*

1.8.13 See the results

That is it, launch Insyght (see section "Launch Insyght") to see the results!

2 Other information

2.1 If the web application feels sluggish

The web application has been designed to be fast and responsive regardless of the numbers of genomes in the database. To reinitialize the indexes and reclaim disk space, paste the following commands in a Terminal:
vacuumdb --full --analyze -h localhost -p 5432 -U origami_admin origami_prod

~~ *Remark: safely ignore the warnings on the terminal.*

2.2 Changing the threshold parameters for the programs blast (used to find gene similarity) and align (used to compute the synteny)

Prior to running the step “Add your own data in the database”, you can configure the blast threshold parameter (e-value). Open the file BlastConfig.pm and change the threshold value to your liking (default is -thresh => '5e-2').
vim ${SCRIPTS_DIR}/BlastConfig.pm

Synteny regions are computed with a dynamic programming algorithm called “align” to determine the highest scoring paths among the chain of collinear homologs. By default, small gaps are allowed within the conserved synteny. You can customize those parameters during the step “Syntenies generator”. This script supports the following arguments:
-MAIN_ALIGN_OPTION_os*: defines the ortholog (BDBH) score; default value = 4; must be integer > 0.*
-MAIN_ALIGN_OPTION_hs*: defines the homolog (non-BDBH but significative blast result) score; default value = 2; must be integer > 0.*
-MAIN_ALIGN_OPTION_mp*: defines the mismatch penalty score within a synteny; default value = -4; must be integer < 0.*
-MAIN_ALIGN_OPTION_gc*: defines the gap creation score within a synteny; default value = -3; must be integer < 0. Must be different than the mismatch penalty score due to a current bug.*
-MAIN_ALIGN_OPTION_ge*: defines the gap extension score within a synteny; default value = -3; must be integer < 0. Must be identical to the gap creation score due to a current bug.*
-MAIN_ALIGN_OPTION_m*: defines the minimum size for a synteny (number of CDS); default value = 1; must be integer > 0.*
-MAIN_ALIGN_OPTION_c*: defines the cutoff score for a synteny; default value = 8; must be integer > 0.*
-MAIN_ALIGN_OPTION_o*: defines if the program must include all orthologs regardless of the cutoff score; default value = 1; must be integer 1 (Yes) or 0 (No).*

-MAIN_ALIGN_OPTION_pf: *defines the minimum protein fraction for Ortholog; default value = 0.5; must be double between 0 and 1.*

For example, here is a command that uses two of the optional arguments above:
perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
**Task_add_alignment_generator_for_IDRIS.pl -MAIN_ALIGN_OPTION_os 8 **
-MAIN_ALIGN_OPTION_hs 4

~~ Remark: As you can add data incrementally, it is possible (although not advised) to insert data computed with 2 different sets of blast or align parameters. Comparing the results between data generated with different sets of parameters is strongly discouraged however. Running the step "Check the consistency of the database", will raise a warning that the database is deemed not stable.

2.3 Different ways to download a raw fasta or annotated genome file from the ncbi or ebi repositories

2.3.1 Browse and download http content with a web browser

- ncbi http server : http://www.ncbi.nlm.nih.gov/guide/genomes-maps/
- ebi http server : http://www.ebi.ac.uk/genomes/bacteria.html or http://bacteria.ensembl.org/index.html

~~ Remark: There is no web browser installed on the IFB VM but you can use the web browser on your host machine to download the files and then transfer them to the IFB VM using the command provided in the dashboard.

~~ Remark: Upon download on your host machine, the files will be stored in the default "Downloads" directory associated with the web browser.

2.3.2 Browse and download ftp content with a web browser

- ncbi ftp server : ftp://ftp.ncbi.nih.gov/

~~ Remark: To access the genbank content go to ftp://ftp.ncbi.nih.gov/genomes/genbank/. To access the refseq content go to ftp://ftp.ncbi.nih.gov/genomes/refseq/. The ncbi refseq repository is less exhaustive than genbank but contains only annotated genomes.

~~ Remark: To search a particular word within the web page, use ctrl-f. You can use this trick to find your genome of interest among the long list of organisms.

~~ Remark: Prefer the latest assembly for a given organism (directory latest_assembly_versions). The .fna files are the raw fasta file, the .gbff files are the annotated genome files (if any).

- ebi ftp server : ftp://ftp.ensemblgenomes.org/pub/bacteria/current

~~ Remark: At the time of this writing, the ebi ftp server allows for download of the fasta files but not the annotated genome files in embl or genbank format.

2.3.3 Browse and download ftp content with a terminal

You also have the possibility to download files from a terminal connected to the IFB VM using the ftp server as follow:
Change directory on your VM to the “Downloads” directory, for example:
cd /root/mydisk/
Connect to the ftp repository, for example:
ftp ftp.ncbi.nih.gov
Log in to the system:
Name: **anonymous**
Password: **** YOUR EMAIL ADDRESS ****

~~ Remark: the ftp directories structure is similar using a terminal or a web browser.

To navigate among the ftp directories structure, you can use the **cd** command. For example:
cd genomes/genbank
To list all the files in a directory:
ls
Use the wildcard **** to search for your organism of interest inside a directory, for example:
ls *Acetohalobium*

~~ Remark: Not all ftp servers support all commands due to performance issues. You will get an error “504 Command not implemented for that parameter” if the command is not implemented.

To read a file in the terminal via ftp, use **get **FILE** /dev/tty**. For example:
get README.txt /dev/tty
To download the file on your VM, use the **bin** and **get **FILE**** commands. For example:
bin
get GCA_000144695.1_ASM14469v1_genomic.fna.gz
get GCA_000144695.1_ASM14469v1_genomic.gbff.gz
Log out of the ftp session:
bye
You should be back to your default shell session. List and decompress the files you just downloaded. For example:
ll

gunzip GCA_000144695.1_ASM14469v1_genomic.fna.gz
gunzip GCA_000144695.1_ASM14469v1_genomic.gbff.gz
Now you can visualize the content of the file:
less GCA_000144695.1_ASM14469v1_genomic.fna
less GCA_000144695.1_ASM14469v1_genomic.gbff
To proceed with the pipeline, move the annotated genomes files (gbff and gbk) in the **${BANK_DIR}** directory (see section "Copy all the .gbk, .dat and .gbff genomes files you want to add"):
mv GCA_000144695.1_ASM14469v1_genomic.gbff ${BANK_DIR}
If no annotated genome files are available, you can annotate the raw fasta (.fna or .fa) files with prokka (see section "Run prokka to quickly annotate raw fasta file").

2.4 Run prokka to quickly annotate raw fasta file

Prokka (http://www.vicbioinformatics.com/software.prokka.shtml) is a software tool for the rapid annotation of prokaryotic genomes developed by Victorian Bioinformatics Consortium. It is pre-installed on this virtual machine. The following commands delete any previous output and run prokka on the example file:

rm -f -r /root/mydisk/prokka_output/*
**prokka --compliant --addgenes --mincontiglen 200 **
**--outdir /root/mydisk/prokka_output **
**--genus Acetohalobium --species arabaticum --strain DSM_5501 **
**--centre JOUY --locustag AarD **
**--force **
/root/documents/example_files/fasta/CP002105.fna

To annotate your own fasta file, just substitute the example filename above with the path to your file and adapt the different arguments of prokka (--genus, --species etc.) accordingly.

~~ Remark: Information will be printed on the screen while the script run. If the script is successful, you should see a list of output files followed by a reference to the prokka paper printed on the last lines. To see the resulting files generated by prokka, type:
ll /root/mydisk/prokka_output
The file that we are interested in is the .gbk file; to display it, type:
less /root/mydisk/prokka_output/*.gbk

~~ Remark on the other prokka arguments: type:
prokka -help

~~ Benchmark: This step took ~10 minutes for the fasta file provided as demo with 1 CPU in use.

2.5 Using the IFB cloud cluster (external processors)

Please refer to the latest IFB documentation on how to setup a virtual cluster on the IFB cloud. At the time of this writing, the setup is as follow:

First, enable ssh agent in a terminal on your host computer:
ssh-agent
ssh-add ~/.ssh/id_rsa (or **id_dsa**)

From the IFB dashboard, run a "Bacterial genomics" VM with the virtual disk; it is the master. Run several other "Bacterial genomics" VMs without any vDisk; they are the nodes.

In a terminal, connect to the master VM:
ssh -A ...

Configure the master to register the IP of the nodes:
vim /root/cluster/nodes.l
Add 1 node IP per line as follow:
node-1 IP
…
node-n IP

~~ Remark: You can get the list of IPs from the IFB cloud dashboard by selecting the nodes VMs and clicking on the button 'Get IPs'.

~~ Remark: A tutorial on vim is beyond the scope of this document. As a reminder, press "i" to enter the insert mode, "esc" to return to the command mode, and "wq" to save-exit.

Run the configuration:
ifb-cluster-sge /root/cluster/nodes.l 2

Wait a few minutes until the setup finishes. You might now be logged in as sge-admin (the user you are logged in is visible in the prompt). If this is the case, logout to be root for now:
exit

At this point the setup of the virtual cluster on the IFB cloud for this VM is completed. It will be active until you shut down or terminate the master VM or the nodes.

To be able to submit jobs to the cluster, you must change user to sge-admin.
First, setup the correct files permissions to the relevant directories of the pipeline depending on the step you want to parallelize:

** For the step Task_blast_all_launch_cluster_blast_IDRIS.pl:

mkdir -p ${DATA_DIR}/Task_blast_all/tmp/blast_output
chmod -R 777 ${DATA_DIR}/Task_blast_all/tmp/blast_output
mkdir -p ${DATA_DIR}/Task_blast_all/tmp/blast_done
chmod -R 777 ${DATA_DIR}/Task_blast_all/tmp/blast_done
mkdir -p ${LOG_DIR}/Task_blast_all/launch_blast/
chmod -R 777 ${LOG_DIR}/Task_blast_all/launch_blast/
touch ${LOG_DIR}/Task_blast_all/
Task_blast_all_launch_cluster_blast_IDRIS.log
chmod 777 ${LOG_DIR}/Task_blast_all/
Task_blast_all_launch_cluster_blast_IDRIS.log

** For the step Task_add_alignment_launch_jobs.pl:

mkdir -p ${DATA_DIR}/Task_add_alignments/tmp
chmod -R 777 ${DATA_DIR}/Task_add_alignments/tmp
mkdir -p ${LOG_DIR}/Task_add_alignments/
chmod -R 777 ${LOG_DIR}/Task_add_alignments/

~~ Remark: if you skip this step, you might get errors when submitting your jobs such as "error: can't open output file "XXXX": Permission denied".

Change user to sge-admin:
su - sge-admin

Change your working directory to /root/mydisk:
cd /root/mydisk

~~ Remark: If you skip this step, you might get error messages such as "can't chdir to /ifb/origami/pipeline_origami/ORIGAMI/SCRIPTS: No such file or directory".

Delete the current nohup output file if any:
rm -f nohup.out

You are now ready to use the arguments **-MAX_JOBS** and **-CMD_CLUSTER**. Here are some example commands with **-MAX_JOBS 16** for the 2 steps that support the use of the cluster:

** For the step Task_blast_all_launch_cluster_blast_IDRIS.pl:

nohup perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
**Task_blast_all_launch_cluster_blast_IDRIS.pl **
**-MAX_JOBS 16 -CMD_CLUSTER "qsub -m ea" -EMAIL_NOTIFICATION **
****YOUR_EMAIL_ADDRESS@PROVIDER****

** For the step Task_add_alignment_launch_jobs.pl:

nohup perl -I ${SCRIPTS_DIR}/ ${SCRIPTS_DIR}/
**Task_add_alignment_launch_jobs.pl **
**-MAX_JOBS 16 -CMD_CLUSTER "qsub -m ea" -EMAIL_NOTIFICATION **
****YOUR_EMAIL_ADDRESS@PROVIDER****

~~ Remark: Make sure to customize the -MAX_JOBS option to the number of nodes in your cluster. To know the number of nodes in your cluster, check the IFB dashboard and add up all the CPU of the node VMs. Make sure to customize the -EMAIL_NOTIFICATION to your email address as well.

You will receive an email to notify you when the step is completed. Make sure you provided a correct email address following the option -EMAIL_NOTIFICATION above.