

Directory Structure Maintenance for PGDB

- Once a user uploads sequence files to a sample, the system will create the appropriate folder structure according to a predefined path stated in a system configuration file.
- Initially the system will create a folder with the logged in users' ID. All the fastq files uploaded to the system by a given user will get stored in this specific folder along with the sample folder which those fastq files belongs to.

Here the sample name needs to be unique. No 2 samples can have the same name.

Ex: Assume the predefined path is /data/pgdb/users and user ID is 1001. The folder structure will get created as follows.

/data/pgdb/users/1001/08-5578-small/08-5578_S1_L001_R1_001.fastq

/data/pgdb/users/1001/08-5578-small/08-5578_S1_L001_R2_001.fastq

**** 08-5578-small** : sample folder name

- For every *fastq* file upload, a corresponding record will get inserted to the *sequence_files* table.

id	file_path	created_date
1	/data/pgdb/users/1001/08-5578-small/08-5578_S1_L001_R1_001.fastq	2017-06-23 11:38:03
2	/data/pgdb/users/1001/08-5578-small/08-5578_S1_L001_R2_002.fastq	2017-06-23 11:38:12

- And to the *samples_sequence_files* table (records relationship between samples and sequence files)

id	sample_id	sequence_file_id
1	S000001	1
2	S000001	2

- If the user deletes the uploaded files it will get removed from the DB as well as from the physical path.
- When a pipeline job is executed, the system will perform the following tasks.
 - Running the Trimmomatic tool against the selected fastq files.
 - Creation of the input folder '*mTB_pipeline_inputs_YYYYMDD_hhmmss*'.
 - Transferring the trimmed paired results to '*mTB_pipeline_inputs_YYYYMDD_hhmmss/paired*' folder (this will be the input file folder for the python script)
 - While trimmed unpaired will get transferred to '*mTB_pipeline_inputs_YYYYMDD_hhmmss/unpaired*' folder.
 - Creation of an output folder '*mTB_pipeline_outputs_YYYYMDD_hhmmss*' to store final results of the pipeline. (this will be the output file folder for the python script)
 - Creation of a log file in order to track the progress of the pipeline. (this will be the log file for the python script)

- A json file with pipeline parameters. (this will be the parameter file for the python script)

**** All these files/ folders will get created under the user folder as shown below.**

/data/pgdb/users/1001/mTB_pipeline_inputs_20170629_114938

/data/pgdb/users/1001/mTB_pipeline_inputs_20170629_114938/paired

/data/pgdb/users/1001/mTB_pipeline_inputs_20170629_114938/unpaired

/data/pgdb/users/1001/mTB_pipeline_inputs_20170629_114938/40052_20170629_114938.text

/data/pgdb/users/1001/mTB_pipeline_inputs_20170629_114938/customizeParams_20170629_114938.json

/data/pgdb/users/1001/mTB_pipeline_outputs_20170629_114938

**** At the end of the process the output folder will get compressed (.tgz) in order to make it downloadable through the system.**

- *pipeline_jobs* table will keep track of the above data.

id	User_id	name	type	file_name	file_path	status	Input_path	Output_path	Created_date
1	1001	mTB_pipeline_2017622	mTB_pipeline	1001_20170622_222656.txt	/data/pgdb/users/1001/1001_20170622_222656.txt	0	/data/pgdb/users/1001/mTB_pipeline_inputs_20170629_112434	/data/pgdb/users/1001/mTB_pipeline_outputs_20170629_112434.tgz	2017-06-29 11:25:17

user_id: user who initiated the job

name: pipeline name given by the user

type: original name of the pipeline

file_name: log file name

file_path: log file path

status: records the progress of the pipeline

[Status 0: submitted/ Status 1: in execution/ Status 2: successful / Status -1: failure / Status 3: downloaded]

**** The system will read the progress of the executed pipeline from the log file and updates the status field.**

input_path: pipeline input path (trimmed paired fastq files)

output_path: compressed pipeline output path (where pipeline results get stored)

created_date: timestamp of the record insertion to the db