Use UUID in Pipeline

Xuyang Yuan

2023-12-29

Table of contents

1 Why uuid?

The UUID is needed for both the **filesystem** and **Maestro/Steps**.

1. **Filesystem**
    1. Unique folder name for each sequencing run of a sample or even for each demultiplexing of a sequencing run of a sample, *e.g.*

       `Diag-wgs123-HG12345678-PM-d303c3b866434b1e93fa4aab17280803`
    2. Unique folder name for each analysis (basepipe, triopipe or annopipe). Multiple run of the same analysis are possible.
2. **Maestro/Steps**

    i) UUID for samples:
    a. `*.fastq.gz` files inherit the UUID of their sample, with a "`_R1`" or "`_R2`" suffix. This serves as the identifier for the output of *DotSampleFileWatcher* and the inputs of the *BasepipeRunner* process.
    b. `*.g.vcf` files inherit the UUID of their sample, with a "`_gvcf`" suffix. This serves as the identifier for the output of the basepipe process and the inputs of the triopipe process.
    c. `*.final.vcf` files inherit the UUID of their sample, with a "`_vcf`" suffix. This serves as the identifier for the output of the basepipe process and the inputs of the annopipe process.
    d. `*.final.vcf` files inherit the UUID of proband sample, with a "`_vcf`" suffix. This serves as the identifier for the output of the triopipe process and the inputs of the annopipe process.

    ii) UUID for basepipe, triopipe, and annopipe.
    a. The samples list in `.analysis` file contains the sample names with UUIDs. *e.g.*

```
{
  "samples": [
    "Diag-wgs1-HG12345678-PM-d303c3b866434b1e93fa4aab17280803",
```

```
    "Diag-wgs1-HG12345679-FM-8d72b72c6bc648ab824dc9a5588b8247",
    "Diag-wgs1-HG12345680-MK-d987c0508a6a4e5aa5771e277a07b3ad"
  ]
}
```

b. The pedigree object in a triopipe `.analysis` file contains samples with UUIDs. *e.g.*

```
{"pedigree": {
 "proband": {
   "sample": "Diag-wgs1-HG12345678-PM-d303c3b866434b1e93fa4aab17280803"
 },
 "father": {
   "sample": "Diag-wgs1-HG12345679-FM-8d72b72c6bc648ab824dc9a5588b8247"
 },
 "mother": {
   "sample": "Diag-wgs1-HG12345680-MK-d987c0508a6a4e5aa5771e277a07b3ad"
 }
}}
```

2 How to create uuid?

We are considering that **it's lims-exporter's responsibility** to create uuid for both samples and analyses.

We can use the uuid library in python, *e.g.*

```
import uuid
smp_uuid = uuid.uuid4()
print(smp_uuid)
```

```
1844be2e-403f-4df8-a1d6-a3c44d3fd28c
```

3 UUID in dot sample file

A `.sample` json file will have a field called `uuid` with a uuid4 value, *e.g.*

```
{
  "uuid": "d303c3b866434b1e93fa4aab17280803"
}
```


4 UUID in sample folder

A sample folder structure looks like:

```
Diag-wgs1-HG12345677-0d62311847414b1899e49c36477b7d7c
├── Diag-wgs1-HG12345677.sample
├── Diag-wgs1-HG12345677.taqman
├── HG12345677-Nett-KIT-wgs_S11_R1_001_fastq.gz
└── HG12345677-Nett-KIT-wgs_S11_R2_001_fastq.gz
```

- sample folder name has uuid
- `.sample` file has **no** uuid
- `.taqman` file have **no** uuid
- `.fastq.gz` files have **no** uuid

When checking if a sample folder has already been created, lims-exporter **ignores the UUID part** of the sample folder name.

5 UUID in analyses-work folder

6 UUID in dot analysis file

A `.analysis` will have multiple fields with uuid4 values, *e.g.*


```
{
  "name": "Diag-wgs1-HG12345678-PM-DR-TRIO-74b039f45d0d46519311407de17d5ca1",
  "params": {
    "pedigree": {
      "proband": {
      "sample": "Diag-wgs1-HG12345678-PM-d303c3b866434b1e93fa4aab17280803"
      },
      "father": {
        "sample": "Diag-wgs1-HG12345679-FM-8d72b72c6bc648ab824dc9a5588b8247"
```

```
      },
      "mother": {
        "sample": "Diag-wgs1-HG12345680-MK-d987c0508a6a4e5aa5771e277a07b3ad"
      }
    },
  },
  "samples": [
    "Diag-wgs1-HG12345678-PM-d303c3b866434b1e93fa4aab17280803",
    "Diag-wgs1-HG12345679-FM-8d72b72c6bc648ab824dc9a5588b8247",
    "Diag-wgs1-HG12345680-MK-d987c0508a6a4e5aa5771e277a07b3ad"
  ]
}
```

7 UUID for reanalyses

When UUIDs are used in `.sample` and `.analysis` files, we need to consider the reanalysis scenario. This is because for reanalyses, instead of creating new UUIDs, the UUIDs of the old samples must be used.

Do we need to make the UUIDs of old samples accessible by lims-exporter?

7.1 No, we don't need to

- ☒ For new analysis, lims-exporter just need to create UUID for new samples and analyses and use them in the `.sample` and `.analysis` files and in the samples and alalyses-work folder names. The UUID will end up in Neo4j database.
- ☐ For reanalysis, we only need the old sample name and project, Neo4j database will be queried to to find the correct UUID of the sample, *e.g.* query by `sample_name` and `project_name` together with other attributes such as `sequencing_run_date`, `sequencer_type` etc.
  - ☐ Who will query Neo4j database, *DotAnalysisFileWatcher*? (there is no `.sample` and `fastq.gz` files for a reanalyzed sample, *DotSampleFileWatcher* or *FastqFileWatcher* will not be triggered.)
  - ☐ How to query Neo4j db in Steps?
  - ☐ Violation of the principle of "Steps needs to know nothing of the Neo4j database"?

7.2 Yes, we need to

- ☐ For reanalysis, we need the path to the sample folder of the old sample on filesystem.
- ☒ If the UUID is not stored, for incomplete trios[1], the sample folder will be created again and again, each time with a new UUID.
  - but this can be avoided; lims-exporter always checks if a sample folder has already been created, letting lims-exporter ignore the UUID part when checking if a sample folder has already been created solves the problem.
- ☒ For the "failed fingerprinting" + "then new SNP-ID file" scenario, we will create a new UUID for the sample which is exactly the same sample.
  - some duplicates in Neo4j database is not a big deal?

If we want to store the UUID of historical samples, we have several options each with pros and cons.

7.2.1 Option 1: Store the UUID as *input sample* UDF[2] in Clarity LIMS.

The lims-exporter checks if the sample has a UUID UDF. If not, it will create one and store it as the sample's UUID UDF in the Clarity LIMS database; if yes, that UUID will be used.

Pros

Pros

1. Can directly query the Clarity database to get old sample name, project and UUID, simple and fast.

Cons

Cons

1. Re-prep, re-sequencing or re-demultiplexing will not be distinguishable becuase the same UUID (from input sample UDF) will be used.

[1] Incomplete trios are trios with missing family members in the lims-exporter step, *e.g.* only proband and mother, or only proband and father, or only proband.
[2] User Defined Field in Clarity LIMS.

7.2.2 Option 2: lims-exporter creates UUID for samples and store the UUID as attribute of the "Demultiplexing" process output file attributes in Clarity LIMS.

Pros

Pros

1. Each re-prep and/or re-sequencing and/or re-demultiplexing will get a new UUID.

Cons

Cons

1. The UUID is stored as the Demultiplexing process's output artifact's UDF, making it hard to retrieve that UUID.
   - ◇ to query the Clarity database, complex query needed, if even possible.
   - ◇ to use the Clarity API, super slow.

7.2.3 Option 3: Maintain a SQLite database with sample name and UUID.

Pros

Pros

1. Querying is fast.
2. SQLite database can be re-created from Neo4j database if needed.

Cons

Cons

1. Adding one more dependency to lims-exporter.

7.2.4 Option 4: Let lims-exporter query Neo4j database to get the UUID of the old sample.

The lims-exporter will query the Claity LIMS database to get the old sample name and old project name, and then query the Neo4j database to get the UUID of the old sample.

The lims-exporter will need other metadata to compose the Neo4j query, *e.g.* sequencing_run_date, sequencer_type, etc.

Pros

Pros

1. Only one source of truth.
2. No need to "write to" Clarity LIMS database by lims-exporter.

Cons

Cons

1. Added a dependency to lims-exporter
   i Neo4j database itself must be accessible by lims-exporter on NSC.
   ii noe4j Python package used by lims-exporter to connect to Neo4j database and to make queries.
2. ~~Circular dependency? (Neo4j database depends on lims-exporter as data source and lims-exporter depends on Neo4j database for UUID.)~~
3. Requires the Neo4j database to be up-to-date.