Samplot-ML on GCP

Author: Giovanni Marchetti (Google)   Revision: 0.1   Date: 04/8/2022

This document is intended to supplement the Google Life Sciences tutorial, with focus on running Samplot-ML.

Samplot-ML (github) is a tool to detect structural variants in genomes. It is derived from Samplot, a utility that will generate diagrams of such variants based on annotation files produced by another classifier (aka a caller). The objective is to reduce the number of false positives without significant impact to the true positives, i.e. to increase the precision, while maintaining a high recall.

Model predictions

Samplot-ML refines the classification by invoking a machine learning model that employs a convolutional neural network. The process requires:

1. A pre-trained Tensorflow model, supplied by the authors in their github repository.
2. A dataset comprised of cropped samplot output images

At the time of writing the model is trained to recognize only deletions.
It produces an output file that contains the locations of such deletions in .bed format. The format is widely recognized by bioinformatics tools.

Using Samplot

Samplot generates the images used as input for the ML algorithm. In turn, it requires:

1. An alignment file (aka a “bam” or “cram” file).
2. A reference genome file (aka a “fasta” file), if using a “cram” file.
3. Type of structural variant. In our case, only deletions apply.
4. Chromosome of interest
5. Start and end position of regions of interest, as obtained from an annotation file (aka a vcf file) and recorded in a “bed” file.
6. The annotation file in question, as generated by a previous caller (e.g. smoove or manta) whose results we want to amend.

Snakemake workflow

To produce all the prerequisite files in the correct order, a workflow tool like snakemake is necessary. It is not the only one available, but it provides native integration with Google Life Sciences pipelines. The pipelines consist of a series of docker containers that are run directly on Google’s cloud infrastructure, thus taking advantage of the resources available there for those compute-and memory-intensive tasks.

A snakemake workflow is a collection of rules, defined according to a template like the following code block:

```
rule <rule name>:
    input:
        <input file 1>,
        <input file 2>,
    output:
        <output file or directory>
    conda:
       <some environment.yaml file>
    resources:
        machine_type=<machine size>,
        mem_mb=<memory required, default=1000>,
        disk_mb=<disk required, default=1000>
    shell:
        "<some command> {input}  {output}"
    # OR instead of shell #
    run:
       <python code snippet {input} {output}>
```

The output of one rule can be the input of another, thus building an execution graph. The snakemake tool will then run the tasks on the graph in the correct order and in parallel when possible.

Workflow steps

A sample snakemake file is provided in https://github.com/gmarchetti2020/samplot-ml. Please clone the repository and start from there.
The workflow for samplot-ml consists of the following steps:

1. Get the reference file and index (“fasta” and “fai”). This can be a download from a public endpoint, or a copy if you already have them.
2. Get the base annotation file (“vcf”) as produced by a caller tool.
3. Generate a region file (.bed) from the annotations (.vcf) for the deletions that have been identified therein. We use the bcftools utility.
4. Generate the images with samplot, using all of the above files plus the alignment file (.bam or .cram) and relative index (.bai or .crai).
5. Zoom and crop the images generated in step 4 to focus on the regions of interest.
6. Produce a final list of the cropped image file locations
7. Classify the images in the list at step 6 using a pre-trained ML model (provided). Store the results in a .bed file.
8. Create a new annotation (.vcf) file based on the classification scores of step 7 and the original vcf file.
9. Download the results.

Each step:

- Runs in a separate container.
- May require different resources (memory, cpus, gpus, machine images) as specified in the resources section of its rule,
- May install different dependencies as per the conda section. If none is specified, then you should create an `environment.yaml` file in your working directory that will apply to all rules.

In editing your workflow file for I/O on cloud storage, remember that:

- Local storage is volatile and wiped out when the container terminates.
- A bucket is not a file system. No folders actually exist (they are just part of the file name), no file there is executable and permissions are assigned with the appropriate GCP roles, not with Unix commands.
- If you need to produce multiple files, snakemake can use directories as output, as long as they are tagged with `directory(<path>)` and created somewhere within the rule (e.g. with `mkdir -p <path>`.
- The files specified in the input section are copied in at the start and those in the output section are copied out to a bucket at the end.

Lifesciences pipelines

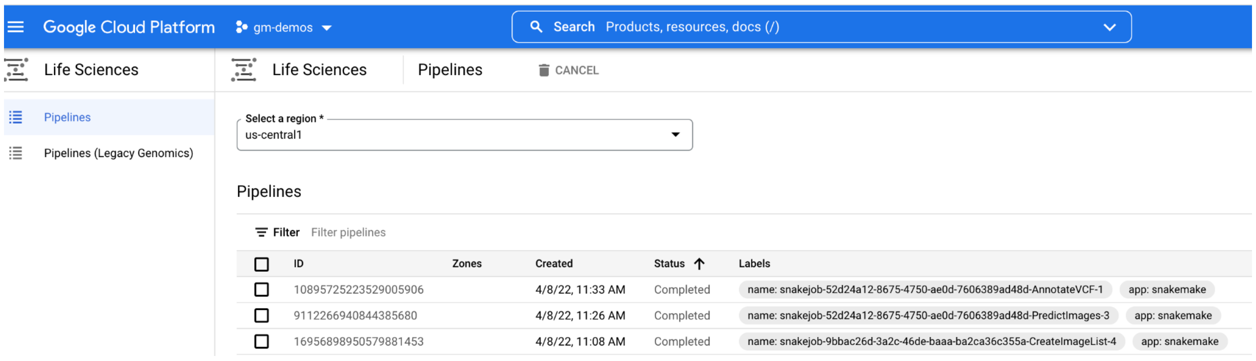
Snakemake supports Google lifesciences pipelines directly as an execution environment. The documentation describes in more detail how to set up a service account, obtain a key file and other key concepts of the GCP integration with the tool.

There are a few parameters to keep in mind, e.g.

```
snakemake \\
--google-lifesciences \\ # the executor environment
--default-remote-prefix <bucket> \\ # the root storage bucket
--use-conda \\ # conda to install prerequisites
--google-lifesciences-region <region> \\ # the region where you run
--jobs <n., e.g. 4> \\ # maximum number of containers to run concurrently
--cores <n., e.g. 8> \\ # maximum number of cores in a machine
-s <workflow file>.smk \\ # the workflow description file
--rerun-incomplete \\ # allow to re-run incomplete steps
--default-resources “mem_mb=1000” “disk_mb=1000” \\ # change defaults
--latency-wait 30 # increase latency for storage operation to 30 seconds
```

When running the command, the content of the current directory is packaged and uploaded to a container that already exists on Google’s registry. The container is pre-configured for snakemake workflows and several bioinformatics workloads.

Once you run the workflow, the GCP console will show the tasks in the pipeline being executed. You will also be able to query operation status end extract logs from the command line.

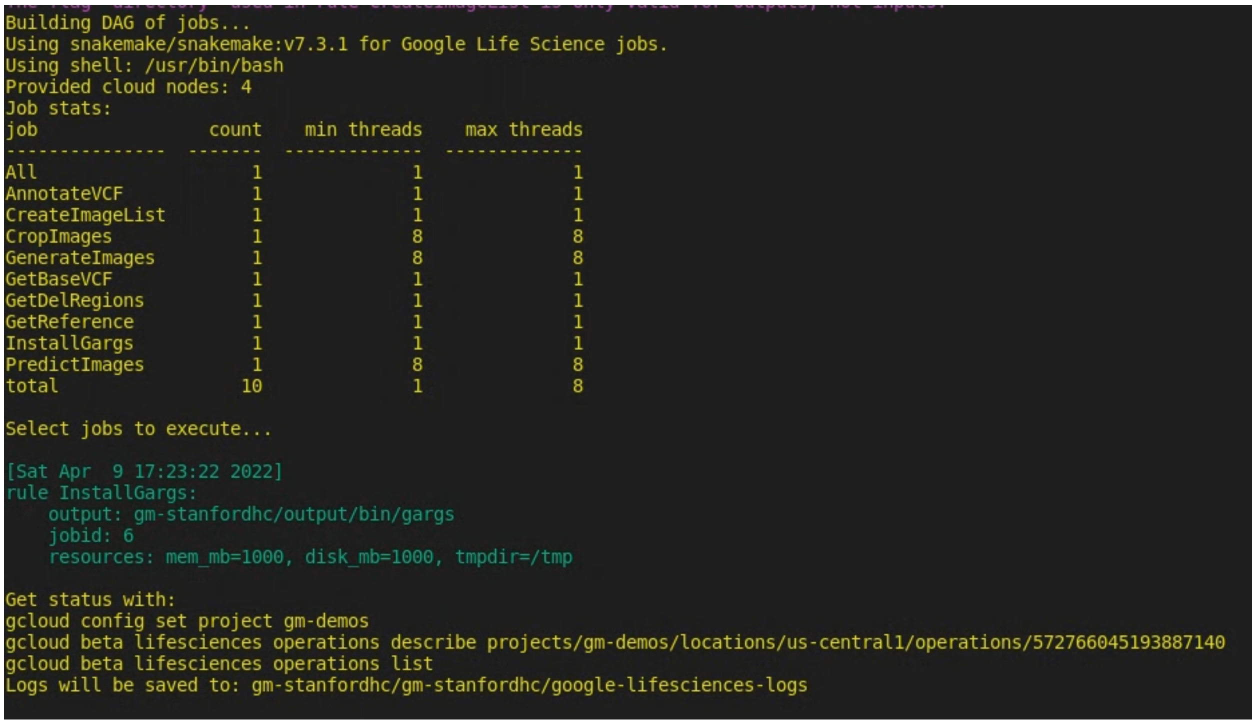


Select a region *
us-central1

Pipelines

Filter Filter pipelines

ID	Zones	Created	Status	Labels
10895725223529005906		4/8/22, 11:33 AM	Completed	name: snakejob-52d24a12-8675-4750-ae0d-7606389ad48d-AnnotateVCF-1 app: snakemake
9112266940844385680		4/8/22, 11:26 AM	Completed	name: snakejob-52d24a12-8675-4750-ae0d-7606389ad48d-PredictImages-3 app: snakemake
16956898950579881453		4/8/22, 11:08 AM	Completed	name: snakejob-9bbac26d-3a2c-46de-baaa-ba2ca36c355a-CreateImageList-4 app: snakemake



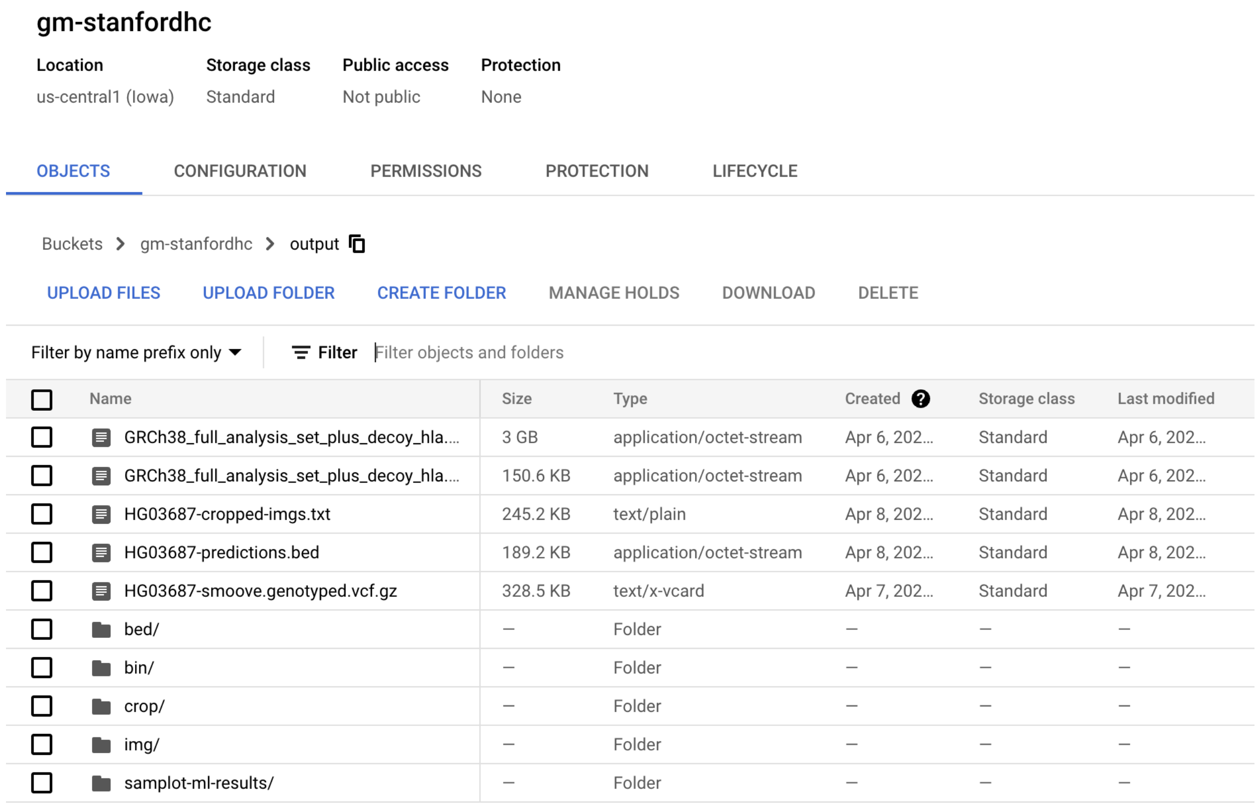
```
Building DAG of jobs...
Using snakemake/snakemake:v7.3.1 for Google Life Science jobs.
Using shell: /usr/bin/bash
Provided cloud nodes: 4
Job stats:
job                count    min threads    max threads
---------------  -------  -------------  -------------
All                    1              1              1
AnnotateVCF            1              1              1
CreateImageList        1              1              1
CropImages             1              8              8
GenerateImages         1              8              8
GetBaseVCF             1              1              1
GetDelRegions          1              1              1
GetReference           1              1              1
InstallGargs           1              1              1
PredictImages          1              8              8
total                 10              1              8

Select jobs to execute...

[Sat Apr  9 17:23:22 2022]
rule InstallGargs:
    output: gm-stanfordhc/output/bin/gargs
    jobid: 6
    resources: mem_mb=1000, disk_mb=1000, tmpdir=/tmp

Get status with:
gcloud config set project gm-demos
gcloud beta lifesciences operations describe projects/gm-demos/locations/us-central1/operations/572766045193887140
gcloud beta lifesciences operations list
Logs will be saved to: gm-stanfordhc/gm-stanfordhc/google-lifesciences-logs
```

Inputs, outputs and logs will be stored in the storage bucket passed as an argument.



gm-stanfordhc

Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Not public	None

☐	Name	Size	Type	Created	Storage class	Last modified
☐	GRCh38_full_analysis_set_plus_decoy_hla....	3 GB	application/octet-stream	Apr 6, 202...	Standard	Apr 6, 202...
☐	GRCh38_full_analysis_set_plus_decoy_hla....	150.6 KB	application/octet-stream	Apr 6, 202...	Standard	Apr 6, 202...
☐	HG03687-cropped-imgs.txt	245.2 KB	text/plain	Apr 8, 202...	Standard	Apr 8, 202...
☐	HG03687-predictions.bed	189.2 KB	application/octet-stream	Apr 8, 202...	Standard	Apr 8, 202...
☐	HG03687-smoove.genotyped.vcf.gz	328.5 KB	text/x-vcard	Apr 7, 202...	Standard	Apr 7, 202...
☐	bed/	—	Folder	—	—	—
☐	bin/	—	Folder	—	—	—
☐	crop/	—	Folder	—	—	—
☐	img/	—	Folder	—	—	—
☐	samplot-ml-results/	—	Folder	—	—	—

References

Belyeu, J.R., Chowdhury, M., Brown, J. et al. Samplot: a platform for structural variant visual validation and automated filtering. Genome Biol 22, 161 (2021). https://doi.org/10.1186/s13059-021-02380-5