Via Technology

EUROCONTROL

Trajectories Pipeline User Guide

Via Technology Ltd.

Trajectories Pipeline User Guide

Phone: +44 1202 708476

Table of Contents

Table of Figures

Revision History

Issue	Date	Description
Draft A	19 August 2018	Initial draft
Draft B	20 August 2018	Incorporate internal review comments.
Draft C	20 September 2018	Update intersection processing guide.
Draft D	21 September 2018	Incorporate internal review comments.
1.0.0	11 December 2018	Raised to issue 1.0.0 for contract close-out.

References

Number	Document
1	Amendment No. 1 to Contract No. 17-110452-C Trajectories Production
2	https://kubernetes.io/
3	https://cloud.google.com/
4	https://www.docker.com/
5	https://cloud.google.com/storage/docs/gsutil

Glossary

Item	Description
CSV	Comma Separated Variable
GCP	Google Cloud Platform
JSON	Java Script Object Notation

Section 1

1 Introduction

1. This document describes how to use the Trajectories Production Pipeline for Amendment No. 1 to Contract No. 17-110452-C Trajectories Production via Cloud-Based Analytics for Performance Monitoring and Review, see [1].
2. The Trajectories Production Pipeline merges trajectory data from different sources then: analyses, interpolates and finds intersections with the merged trajectories, see Figure 1.

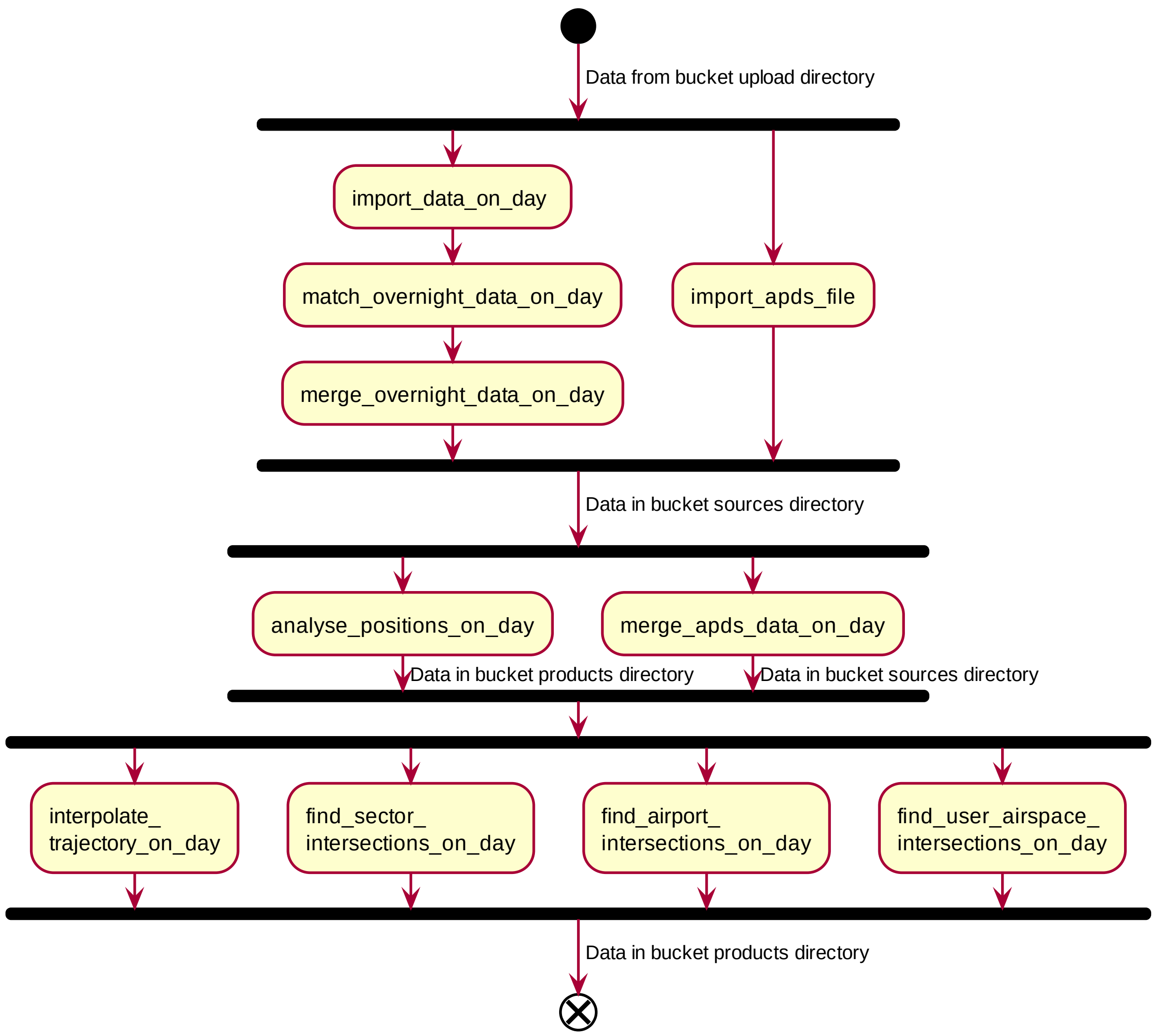

Figure 1: Trajectories Pipeline Processes

3. The Trajectories Production Pipeline runs on a Kubernetes [2] cluster on the Google Cloud Platform (GCP) [3].
4. A key feature of Kubernetes is it’s ability to support “horizontal scaling”, i.e. to run multiple processes in parallel.
5. The processes shown in Figure 1 must be run in the order shown for the same day’s data, e.g.: match_overnight_data_on_day must be run after import_data_on_day and before merge_overnight_data_on_day.
6. However, all the processes shown in Figure 1 can be run in parallel for different days data, e.g. match_overnight_data_on_day can be run in parallel for every day of a week, month, or even a year or more.

Section
2

2 System Setup

7. The Trajectories Production Pipeline runs on the Google Cloud Platform (GCP) [3].

2.1 GCP Project

8. GCP applications operate within the context of a project.

9. A project has a location which determines the geographical location of the associated project resources. The project must be named and a location must be selected.

10. Please note that a Google project has both a name an id and a number. The id may be auto-generated and will not necessarily be the same as the name. The name, id and number can be found on the gcloud console project information screen.

2.2 Google Bucket Storage

11. A Google bucket is used for long term data storage and backup storage.

12. This is associated with a project and must be created through the Gcloud console.

2.3 Kubernetes Cluster

13. The Trajectories Production Pipeline runs on a Kubernetes [2] cluster.

14. This is associated with a project and can be created through the Gcloud console, see Figure 2.

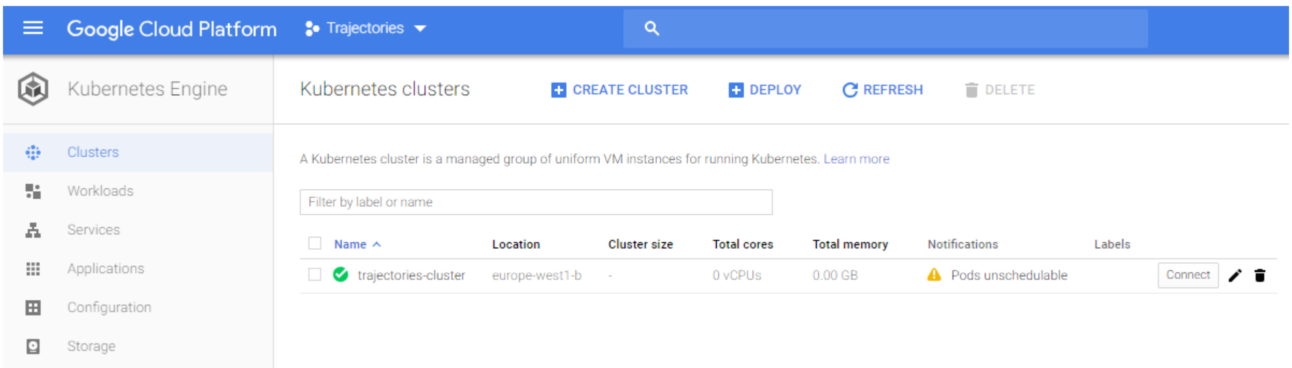

Figure 2: GCP Kubernetes Clusters page

2.4 Pipeline Container Image

15. The Trajectories Production Pipeline processes run on Kubernetes in a Docker [4] container. The docker container must be built and pushed to the relevant container registry for Kubernetes to be able to run the pipeline.

16. The Trajectories Production Pipeline runs a "python-only" container image. The container image is built in two parts:
    - the base image is built locally,
    - then the python code is built from the base image and pushed to the GCP container registry.

2.4.1 Build Base Image

17. The "python-only" base image is built from the BitBucket rt-config repository.

18. Clone the rt-config repository and from the rt-config directory, run:

```
docker build . -f docker/Dockerfile-python-base -t python-base:0.1.0
```

2.4.2 Build Python-only Image

19. The "python-only" image is built from the BitBucket rt-python repository.

20. Clone the rt-python repository and from the rt-python directory, run:

```
docker build . -f docker/Dockerfile-python-rt -t eu.gcr.io/<project>/python-rt:0.1.0
```

21. where <`project`> is the GCP project id, e.g.: shining-booth-205512.

22. To push the image to the GCP container registry, run:

```
docker push eu.gcr.io/<project>/python-rt:0.1.0
```


Section

3

3 Data Organisation

23. The Trajectories Production Pipeline processes read and write data to and from Google Bucket storage.

24. The bucket has the following directory structure:

```
pru-ta
|
|--airports
|  |--stands
|--airspaces
|  |--elementary
|  |--user_defined
|--products
|  |--error_metrics
|  |  |--cpr
|  |  |--cpr_fr24
|  |  |--|--overnight
|  |  |--fr24
|  |--fleet_data
|  |--intersections
|  |  |--airport
|  |  |  |--cpr
|  |  |  |--cpr_fr24
|  |  |  |--fr24
|  |  |-- sector
|  |  |  |--cpr
|  |  |  |--cpr_fr24
|  |  |  |--fr24
|  |  |-- user
|  |  |  |--cpr
|  |  |  |--cpr_fr24
|  |  |  |--fr24
|  |--synth_positions
|  |  |--cpr
|  |  |--cpr_fr24
|  |  |--fr24
|  |--traj_metrics
|  |  |--cpr
|  |  |--cpr_fr24
|  |  |--fr24
|  |--trajectories
|  |  |--cpr
|  |  |--cpr_fr24
|  |  |--fr24
|--refined
```



```
|   |--apds
|   |--cpr
|   |--fr24
|   |--merged
|   |   |--apds_cpr_fr24
|   |   |--daily_cpr_fr24
|   |   |--overnight_cpr_fr24
|--upload
|   |--apds
|   |--cpr
|   |--fr24
```

25. Raw data must be copied into the relevant upload bucket before it is processed.

3.1 Bucket Storage Browser

26. The Google Buckets can be managed using the GCP storage browser, see Figure 3.

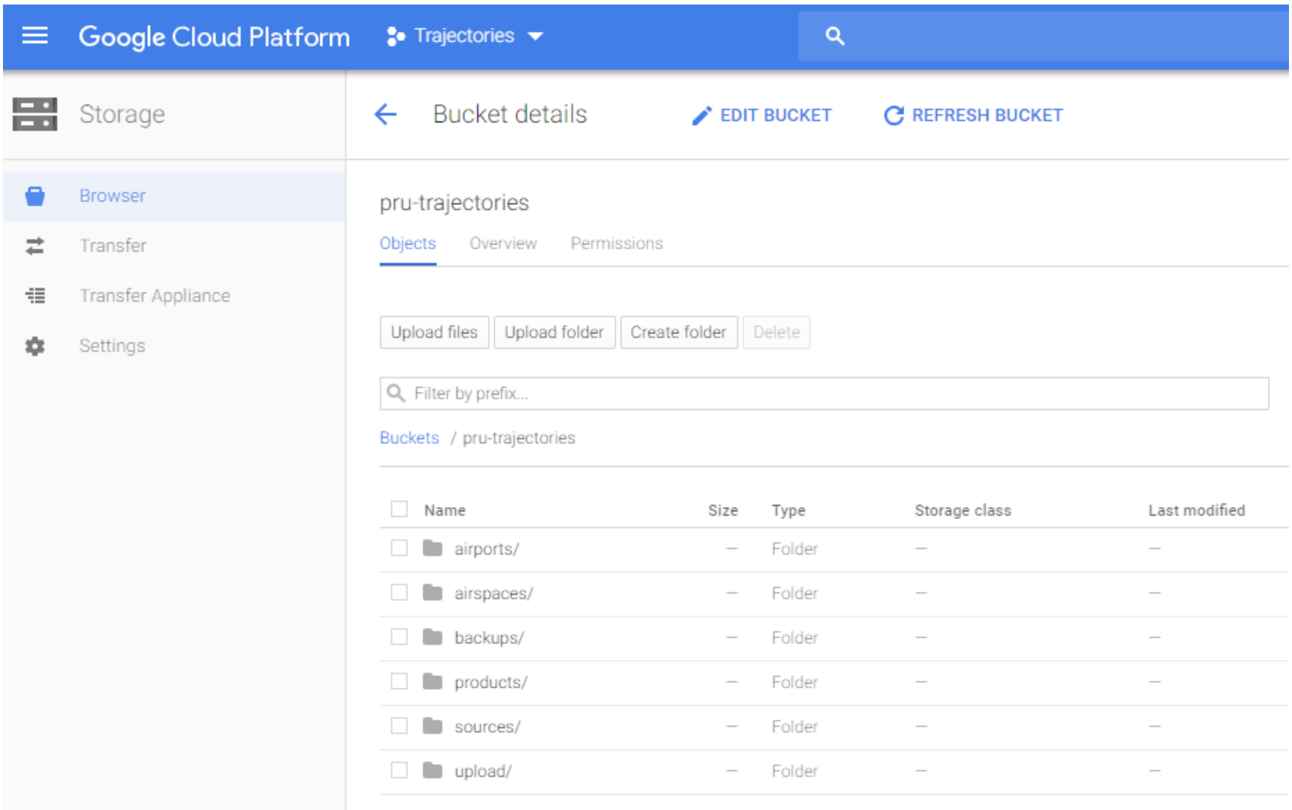

Figure 3: GCP Storage Browser

3.2 gsutil

27. Copying data to and from Google Buckets with the GCP storage browser is a relatively slow process for many large data files.

28. The gsutil tool [5] is a python application that lets you access GCP buckets from the command line, enabling multiple data files to be copied using scripts.

Section

4

4 Pipeline Processes

29. All the Trajectories Production Pipeline processes (see Figure 1) are run as kubernetes jobs.

30. However, most of the processes have different requirements to each other. I.e. some processes require files or even databases to be available, while some processes require significantly more computing resources (i.e. memory and/or cpu) than others.

31. The computing resources required by a process determines which GCP compute nodes the process can run on and how many processes kubernetes can run simultaneously as jobs. The less computing resources required by a process, the more jobs that can scaled horizontally by kubernetes, i.e. run in parallel.

4.1 Importing Daily Data

32. The **import_data_on_day** process refines and merges CPR and FR24 data, see Figure 4.

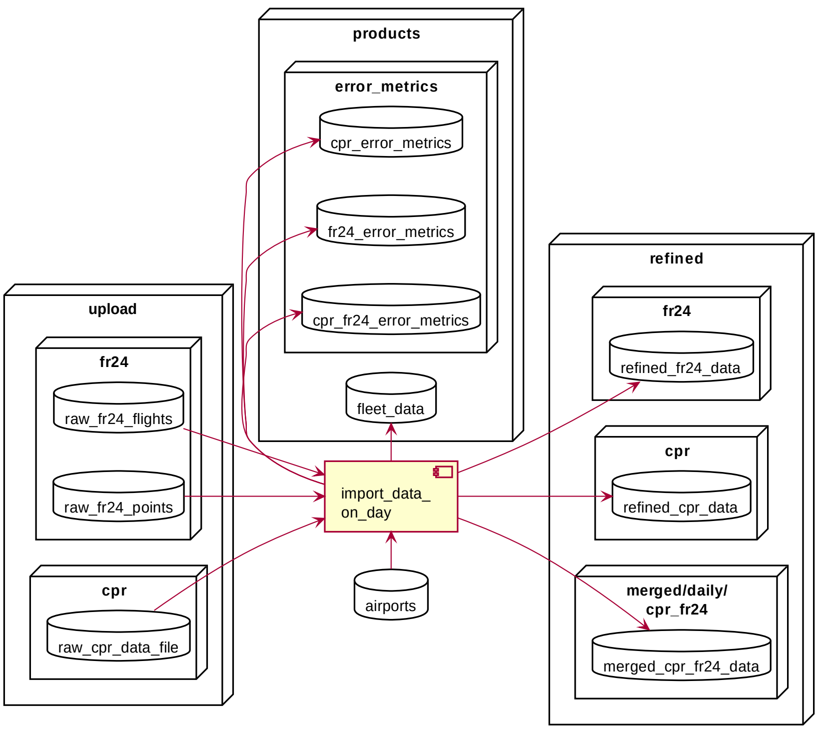

Figure 4: import_data_on_day

33. It runs the following applications:
    - convert_cpr_data.py
    - convert_fr24_data.py
    - convert_airport_ids.py
    - extract_fleet_data.py
    - clean_position_data.py
    - match_cpr_adsb_trajectories.py
    - and merge_cpr_adsb_trajectories.py.
34. To refine and merge CPR and FR24 data for a given day.

4.1.1 Input

35. Data from the bucket upload directory:
    - CPR data from upload/cpr
    - and FR24 data from upload/fr24
36. Airport data from the airports bucket. The default file is airports.csv.

4.1.2 Output

37. Refined data in the bucket sources directory:
    - CPR flights, positions and events files in the bucket sources/cpr directory,
    - FR24 flights and positions files in the bucket refined/fr24 directory,
    - Merged CPR and FR24 flights, positions and events files in the bucket refined/merged/daily_cpr_fr24 directory,
38. Product data in the bucket products directory:
    - fleet data from the FR24 flights in the bucket products/fleet_data
    - error metrics from cleaning the CPR, FR24 and merged CPR_FR24 positions files in the bucket products/error_metrics directory in the cpr, fr24 and cpr_fr24 directories respectively.

4.1.3 Resources

39. This process requires at least 10Gi memory and 1 cpu (note: it can use up to 3 cpus).
40. Each job normally runs to completion in around 1.5 hours.

4.2 Overnight Matching

41. The **match_overnight_flights_on_day** process matches flights for the given day with flights from the previous day and extracts flights that departed on the previous day to be merged later, see Figure 5.

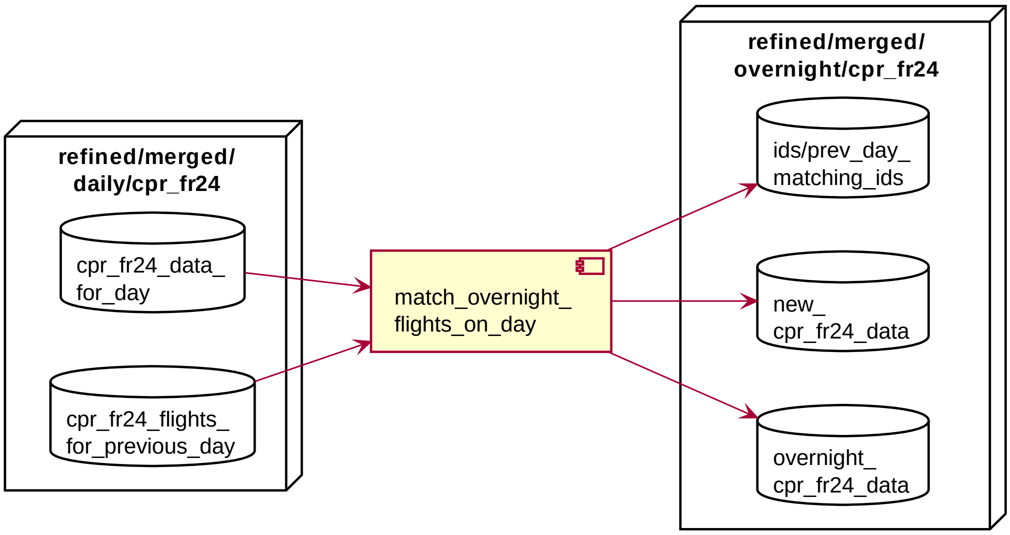

Figure 5: match_overnight_flights_on_day

42. It runs the match_overnight_flights.py and extract_overnight_data.py applications.

4.2.1 Inputs

43. Data from the bucket refined/merged/dailiy_cpr_fr24 directory, i.e.: flights, positions and events

4.2.2 Output

44. Matched data in the bucket refined/merged/overnight_cpr_fr24 directory:
    - overnight positions and events for the previous day
    - and "new" flights, positions and events for the day, without the flights for the previous day
45. Matching flight ids in the bucket refined/merged/overnight_cpr_fr24/ids directory.

4.2.3 Resources

46. This process requires at least 10Gi memory and 1 cpu.
47. Each job normally runs to completion in around 15 minutes.

4.3 Overnight Merging

48. The **merge_overnight_data_on_day** process merges data for the given day with data extracted from the next day by the match_overnight_flights_on_day process, see Figure 6.

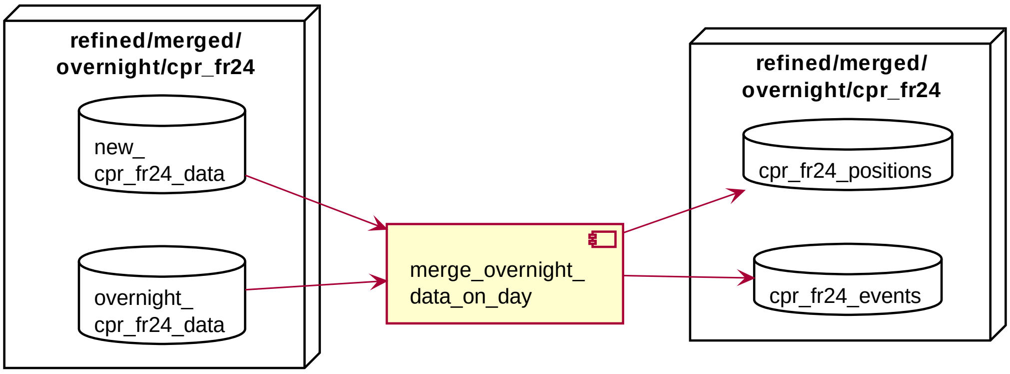

Figure 6: merge_overnight_data_on_day

49. It runs the merge_overnight_flight_data.py and clean_position_data.py applications.

4.3.1 Input

50. Data in the bucket refined/merged/overnight_cpr_fr24 directory:

- the "new" flights, positions and events for the day, without the flights for the previous day
- overnight positions and events from the next day.

51. Note: "new" flights, positions and events for a day are copied into the sources/merged/overnight_cpr_fr24 directory by match_overnight_flights_on_day. Since match_overnight_flights_on_day cannot be run for the first day, the "new" flights, positions and events for first day must be copied from the sources/merged/daily_cpr_fr24 directory with "new_" prepended to their filenames.

4.3.2 Output

52. Merged data for the day in the bucket refined/merged/overnight_cpr_fr24 directory without "new_" or "overnight_" prepended to their filenames.

4.3.3 Resources

53. This process requires at least 10Gi memory and 1 cpu.

54. Each job normally runs to completion in around 40 minutes.

4.4 Trajectory Analysis

55. The **analyse_positions_on_day** process analyses position data for the given day, see Figure 7.

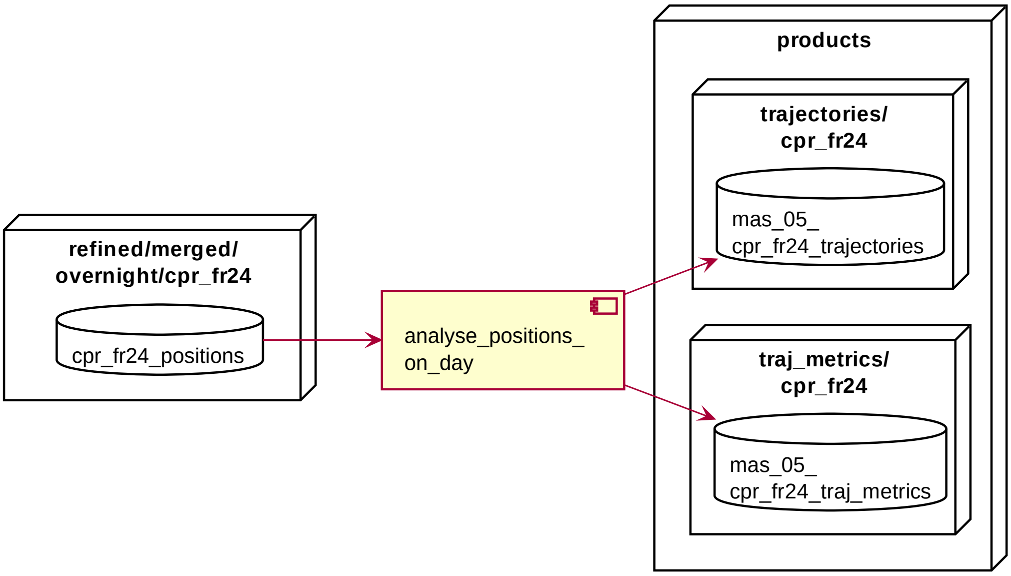

Figure 7: analyse_positions_on_day

56. It runs the analyse_position_data.py application.

4.4.1 Input

57. Merged overnight CPR/FR24 data for the day in the bucket refined/merged/overnight_cpr_fr24.

4.4.2 Output

58. A JSON trajectories file for the day in the bucket products/trajectories/cpr_fr24.

59. A CSV trajectory metrics file in the bucket products/traj_metrics/cpr_fr24.

4.4.3 Resources

60. This process only requires 500Mi memory and up to 1 cpu.

61. Each job normally runs to completion in around 30 minutes.

4.5 Trajectory Interpolation

62. The **interpolate_trajectory_on_day** process interpolates trajectory data for the given day, see Figure 8.

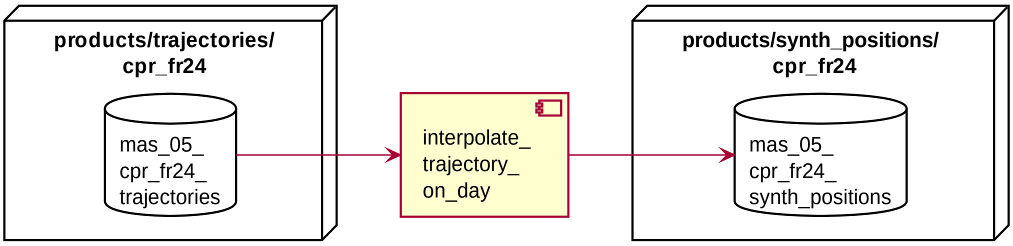

Figure 8: interpolate_trajectory_on_day

63. It runs the interpolate_trajectories.py application.

4.5.1 Input

64. The JSON trajectories file for the day from the bucket products/trajectories/cpr_fr24.

4.5.2 Output

65. A CSV synthetic positions file in the bucket products/synth_positions/cpr_fr24.

4.5.3 Resources

66. This process only requires 500Mi memory and up to 1 cpu.

67. Each job normally runs to completion in around 2 hours.

4.6 Sector Intersections

68. The **find_sector_intersections_on_day** process finds sector intersections for trajectory data for the given day, see Figure 9.

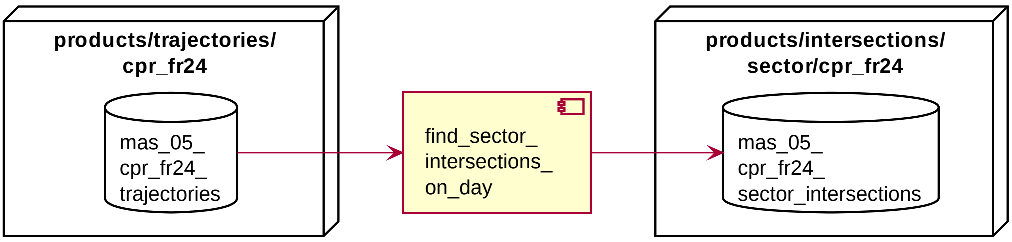

Figure 9: find_sector_intersections_on_day

69. It runs the find_sector_intersections.py application.

4.6.1 Input

70. The JSON trajectories file for the day from the bucket products/trajectories/cpr_fr24.

71. The airspace database must be running and loaded with the airspace sectors.

4.6.2 Output

72. A CSV sector intersections file in the bucket products/intersections/sector/cpr_fr24.

4.6.3 Resources

73. This process only requires 500Mi memory and up to 1 cpu.

74. The process requires the airspace database to be be running, ideally on a compute node with at least one cpu core for each days data, e.g.: an n1-highcpu-32 compute node if being run for an entire AIRAC cycle. Note: the processes can run on the same compute node as the airspace database.

75. Each job normally runs to completion in around 18 hours.

4.7 Airport Intersections

76. The **find_airport_intersections_on_day** process finds airport intersections for trajectory data for the given day, see Figure 10.

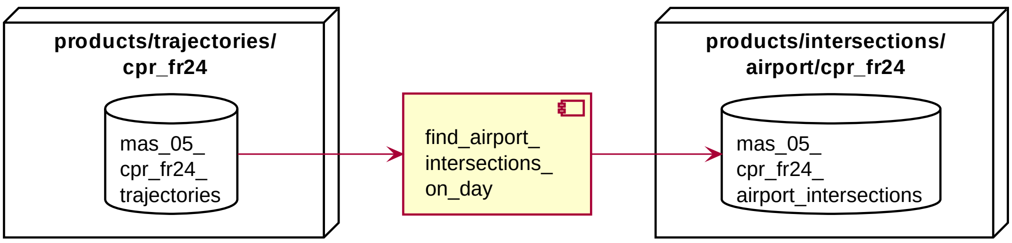

Figure 10: find_airport_intersections_on_day

77. It runs the find_airport_intersections.py application.

4.7.1 Input

78. The JSON trajectories file for the day from the bucket products/trajectories/cpr_fr24.

4.7.2 Output

79. A CSV airport intersections file in the bucket products/intersections/airport/cpr_fr24.

4.7.3 Resources

80. This process only requires 500Mi memory and up to 1 cpu.

81. Each job normally runs to completion in around 15 minutes.

4.8 User Airspace Intersections

82. The **find_user_airspace_intersections_on_day** process finds user defined intersections for trajectory data for the given day, see Figure 11.

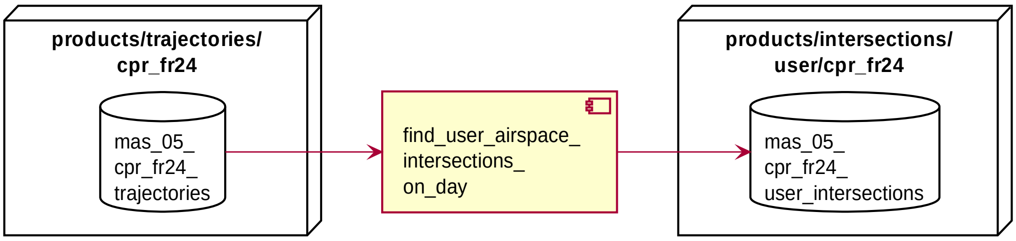

Figure 11: find_sector_intersections_on_day

83. It runs the find_user_airspace_intersections.py application.

4.8.1 Input

84. The JSON trajectories file for the day from the bucket products/trajectories/cpr_fr24.
85. The airspace database must be running and loaded with the user defined airspaces.

4.8.2 Output

86. A CSV user airspace intersections file in the bucket products/intersections/user/cpr_fr24.

4.8.3 Resources

87. This process only requires 500Mi memory and up to 1 cpu.
88. The process requires the airspace database to be be running, ideally on a compute node with at least one cpu core for each days data, e.g.: an n1-highcpu-32 machine if being run for an entire AIRAC cycle. Note: the processes can run on the same machine as the airspace database.
89. The time take for a job to run to completion depends upon the complexity of the airspace.

4.9 Import APDS Data

90. The **import_apds_file** process imports APDS airport data, see Figure 12.

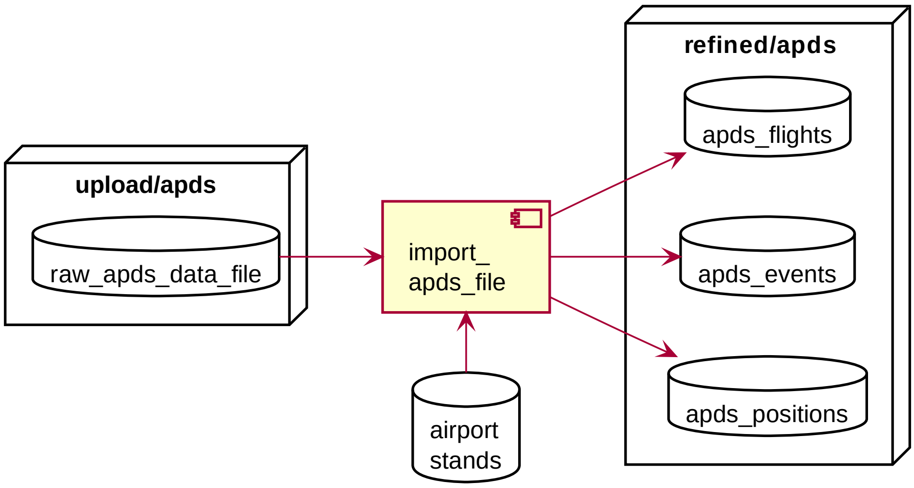

Figure 12: import_apds_file

91. It runs the convert_apt_data.py application.

4.9.1 Input

92. The raw APDS data file, e.g.: FAC_APDS_FLIGHT_IR691_2017-08-01_2017-09-01.csv.bz2.
93. Airport stands data from the airports/stands bucket. The default file is stands_EGLL.csv.

4.9.2 Output

94. APDS flight, event and positions files for the period of the input file, e.g.:
    - apds_flights_2017-08-01_2017-09-01.csv
    - apds_events_2017-08-01_2017-09-01.csv
    - apds_positions_2017-08-01_2017-09-01.csv

4.9.3 Resources

95. This process requires at least 10Gi memory and 1 cpu.
96. Each job normally runs to completion in under 30 minutes.

4.10 Merge APDS Data

97. The **merge_apds_data_on_day** process merges APDS data with the merged

CPR/FR24 data for the given day, see Figure 13.

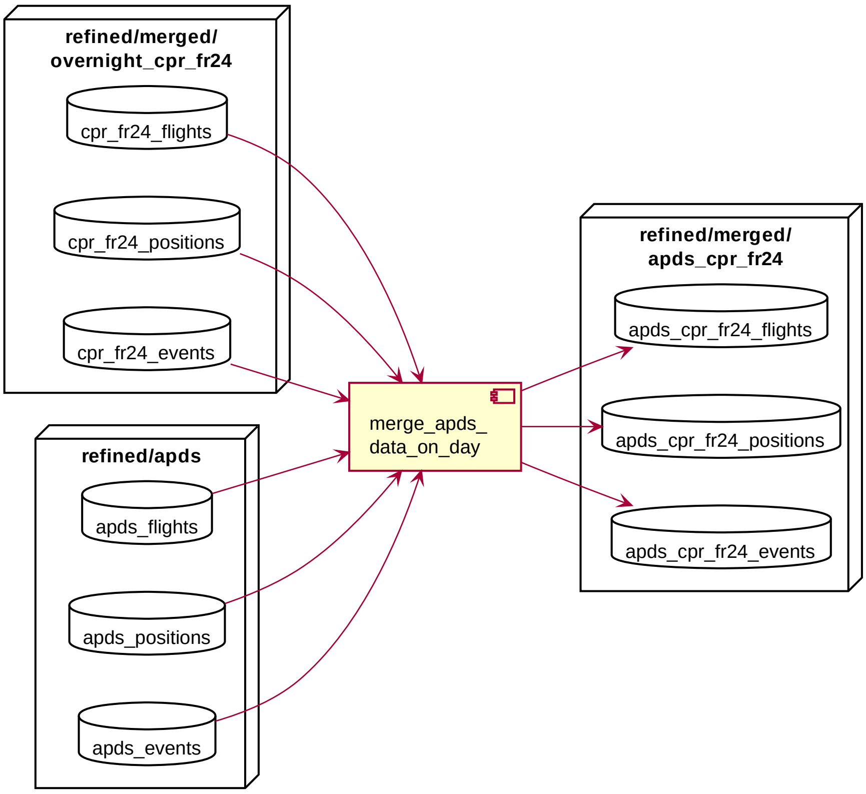

Figure 13: merge_apds_data_on_day

98. It runs the match_apt_trajectories.py and merge_apt_trajectories.py applications.

4.10.1 Input

99. APDS flight, event and positions files for the period of the input file, e.g.:
- apds_flights_2017-08-01_2017-09-01.csv
- apds_events_2017-08-01_2017-09-01.csv
- apds_positions_2017-08-01_2017-09-01.csv

100. Merged CPR and ADSB flight, event and positions files for the given dates.

4.10.2 Output

101. Merged APDS, CPR and ADSB flight, event and positions files for the given dates

4.10.3 Resources

102. This process requires at least 10Gi memory and 1 cpu.

103. Each job normally runs to completion in under 30 minutes.

Section 5

5 Managing Jobs

5.1 Processing Order

104. The CPR and FR24 data must be imported and processed in the order shown in Table 1.

Process	2017-09-01	2017-09-02	2017-09-03	2017-09-04	2017-09-05	2017-09-06	2017-09-07
import_data_on_day							
match_overnight_flights_on_day	See Note 1						
merge_overnight_data_on_day	See Note 2						See Note 3
analyse_positions_on_day							
interpolate_trajectory_on_day							

Table 1: Import Processing Order

105. Table 1 shows which processes can be run in parallel for a weeks data. However, processes may be run in parallel for as many days as input data is available.

106. Note 1: match_overnight_flights_on_day requires data for the date and the *previous* date. Therefore, it cannot be run for the first day in a set of data (unless the days data was imported previously).

107. Note 2: merge_overnight_data_on_day requires data for the date and the *next* date. Since match_overnight_flights_on_day copies data into the merged/overnight/cpr_fr24 bucket for the given day, the data files for the first day in a set of data must be copied over manually to merged/overnight/cpr_fr24 bucket with “new_” prepended to the file names.

108. Note 3: since merge_overnight_data_on_day requires data for the date and the next date, it cannot be run for the last day in a set of data.

5.2 Compute Resources

109. Some processes require more compute resources (memory and cpu) than others. In Table 1, processes with high memory requirements are shown in red, while processes with low memory requirements are shown in green.

110. The high memory processes require at least 10 GB memory (they may require even more memory for short periods). The smallest standard GCP machine type that they can run on is an n1-standard-4 machine with 15GB of memory and 4 cpus.

111. The other processes are limited to 0.5GB and can run on any GCP machine type, including an f1-micro with only 600MB of memory and 0.2 cpu.

112. GCP compute nodes can be run as preemptible nodes: short-lived compute instances that may be shut down with only 30 seconds notice.

113. Preemptible compute nodes are up to 80% cheaper than normal compute nodes and are ideal for running batch jobs like the pipeline processes, see Figure 14.

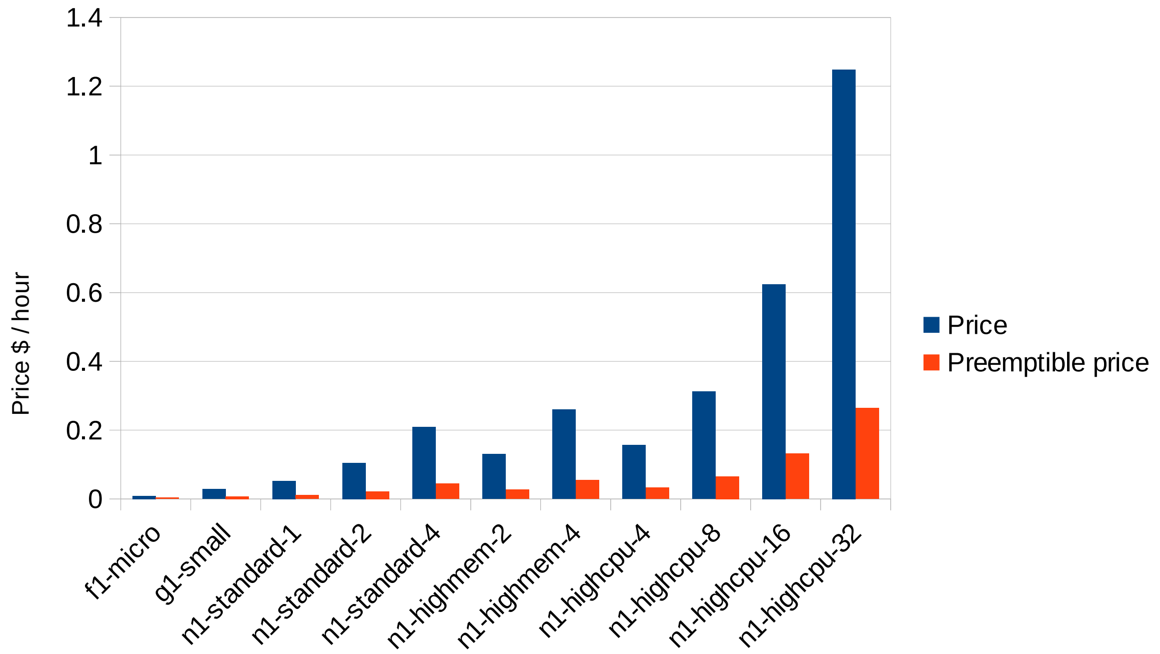

Figure 14: GCP Compute Node Pricing

114. Note: although preemptible f1-micro compute nodes are the cheapest, they are not necessarily the best value, since they only provide 0.2 cpu. For most processes that can run on any GCP machine type, preemptible n1-standard-1 compute nodes theoretically provide the best value. However, despite claiming 1 cpu they only support 0.5cpu...

115. In practice, preemptible g1-small compute nodes or multiple jobs running on a preemptible highcpu compute node provide the best value.

5.2.1 High Memory Processes

116. The processes:

- import_data_on_day,
- match_overnight_flights_on_day,
- merge_overnight_data_on_day,
- import_apds_file
- and merge_apds_data_on_day

117. Each require a compute unit with at least 10Gi memory. The minimum GCP machine type that they can run on as a compute node is n1-highmem-2.

118. However, we recommend running pairs of high memory pipeline processes on n1-highmem-4 compute nodes as the processes can share the 26GB of memory available.

5.2.2 Low Memory Processes

119. The processes:
- analyse_positions_on_day,
- interpolate_trajectory_on_day,
- find_airport_intersections_on_day,
- find_sector_intersections_on_day
- and find_user_airspace_intersections_on_day.

120. Can be limited to 500Mi memory per day. The minimum GCP machine type that they can run on as a compute node is f1-micro. However, the f1-micro normally only receives 0.2 cpu, with a full cpu only allowed in short bursts.

121. Preemptible n1-standard-1 compute nodes provide better value for processes that run at 100% cpu, i.e.: analyse_positions_on_day and interpolate_trajectory_on_day.

5.2.3 Airspace Database

122. The processes:
- find_sector_intersections_on_day
- and find_user_airspace_intersections_on_day.

123. Use the GIS Postgres airspace database to find intersections. Therefore, an instance of the airspace database must be running and loaded with the airspace data for the AIRAC cycle.

124. The airspace database is the performance bottleneck when running these processes. Therefore it is recommended to run it on a “highcpu” node type with at least as many virtual cpus as the number of days being processed.

125. Note: the processes can run on the same compute node as the airspace database. Therefore, the airspace database and find_sector_intersections_on_day can be run for all 28 days of an AIRAC cycle on a preemptible n1-highcpu-32 compute node.

5.3 Running Processes

5.3.1 Provisioning Compute Resources

126. Before running a process or a batch of processes, the kubernetes cluster must be provisioned with enough compute resources (i.e. nodes) to run the processes.

127. The kubernetes master should not run on a preemptible node. A single f1-micro compute node is normally sufficient to run the kubernetes master.

128. Each high memory process requires at least a n1-highmem-2 compute node, or half a n1-highmem-4, a quarter of a n1-highmem-8, etc.

129. Figure 15 shows the configuration of a pool of 12 preemptible n1-highmem-4 compute nodes.

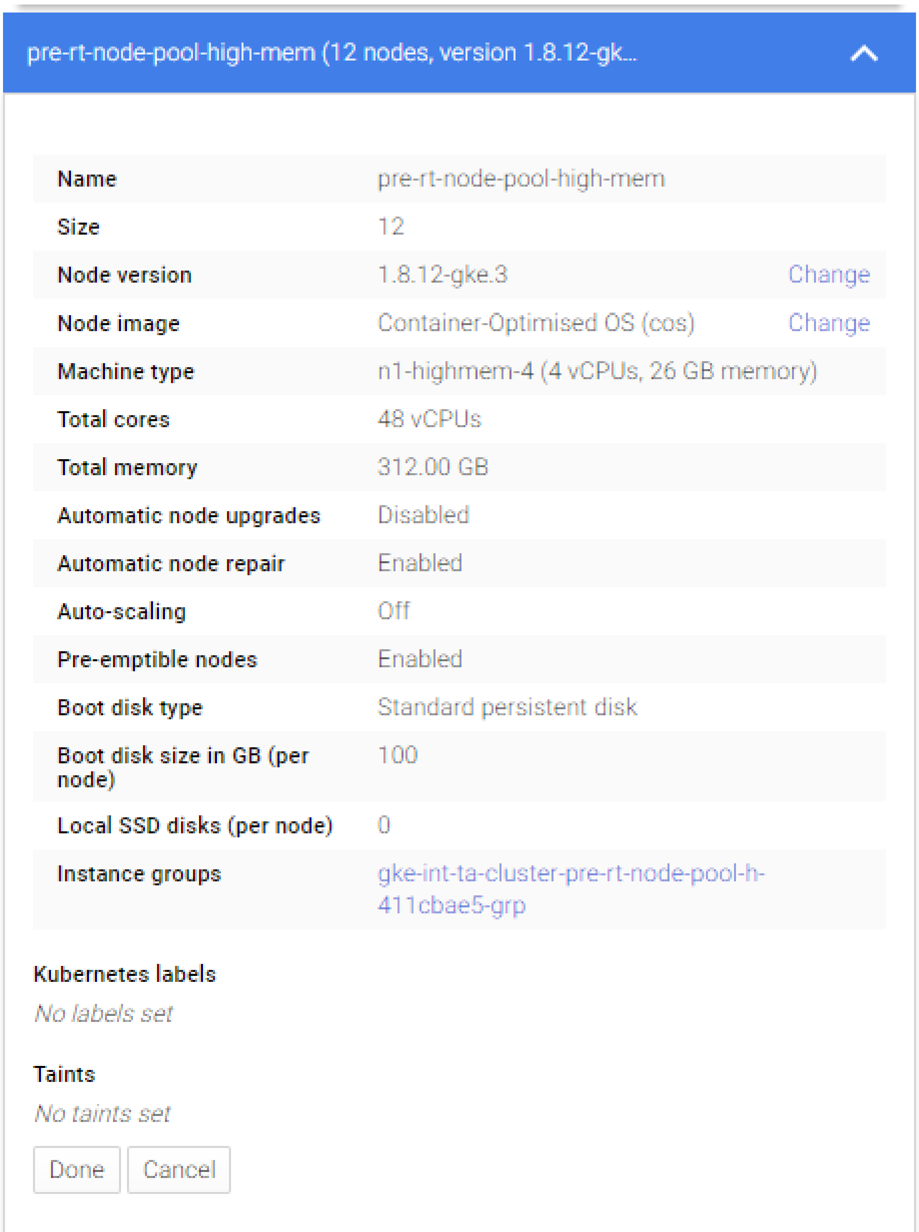

Figure 15: A pre-emptible high-mem node pool

130. Each low memory process can run on a preemptible f1-micro compute node. They will run quite happily on high memory compute nodes, but they are much more expensive, see Figure 14.

131. The exception being the airspace database processes (see section 5.2.3), which require a compute node with enough virtual cpus for each day to be processed.

132. GCP takes around a minute to shutdown each compute node in a node pool: i.e. it is much quicker to shut down a single highcpu compute node than ran multiple jobs than many small or micro compute nodes than ran single jobs. However, some jobs such as interpolate_trajectory_on_day must be run on multiple compute nodes.

5.4 Running Jobs

133. Pipeline processes can be run on the kubernetes compute nodes using the run_kubernetes_jobs.py python script in the rt-config repo scripts directory. It is used as follows:

```
run_kubernetes_jobs <process name> <start date> <finish date>
```

134. where process name is the name of the process: e.g. import_data_on_day and start date and finish date are the first and last dates (in ISO 8601 format) of the data to run the process on.

135. For example:

```
run_kubernetes_jobs.py import_data_on_day 2017-08-07 2017-08-31
```

136. Runs the import_data_on_day process on data for all days between 2017-08-07 and 2017-08-31 inclusive and outputs:

```
('job "pru-import-data-on-day-2017-08-07-job-z8mqt" created\n', None)
('job "pru-import-data-on-day-2017-08-08-job-pzgvt" created\n', None)
('job "pru-import-data-on-day-2017-08-09-job-v8hqj" created\n', None)
('job "pru-import-data-on-day-2017-08-10-job-8vwpz" created\n', None)
('job "pru-import-data-on-day-2017-08-11-job-qprzt" created\n', None)
('job "pru-import-data-on-day-2017-08-12-job-gchvv" created\n', None)
('job "pru-import-data-on-day-2017-08-13-job-6kp7f" created\n', None)
('job "pru-import-data-on-day-2017-08-14-job-f6dcr" created\n', None)
('job "pru-import-data-on-day-2017-08-15-job-qnwxh" created\n', None)
('job "pru-import-data-on-day-2017-08-16-job-rmp8f" created\n', None)
('job "pru-import-data-on-day-2017-08-17-job-9xwpp" created\n', None)
('job "pru-import-data-on-day-2017-08-18-job-vcv8h" created\n', None)
('job "pru-import-data-on-day-2017-08-19-job-hgvtj" created\n', None)
('job "pru-import-data-on-day-2017-08-20-job-9nmdj" created\n', None)
('job "pru-import-data-on-day-2017-08-21-job-qvrcg" created\n', None)
('job "pru-import-data-on-day-2017-08-22-job-2jgrm" created\n', None)
('job "pru-import-data-on-day-2017-08-23-job-jwwk5" created\n', None)
('job "pru-import-data-on-day-2017-08-24-job-7z992" created\n', None)
('job "pru-import-data-on-day-2017-08-25-job-p629k" created\n', None)
('job "pru-import-data-on-day-2017-08-26-job-r4pf4" created\n', None)
('job "pru-import-data-on-day-2017-08-27-job-sr6g4" created\n', None)
('job "pru-import-data-on-day-2017-08-28-job-d6h64" created\n', None)
('job "pru-import-data-on-day-2017-08-29-job-nsdtd" created\n', None)
('job "pru-import-data-on-day-2017-08-30-job-vfq9g" created\n', None)
('job "pru-import-data-on-day-2017-08-31-job-dh8z8" created\n', None)
```

137. Which lists all of the created kubernetes jobs.

138. Note: import_apds_file takes the name of the APDS file, e.g.: FAC_APDS_FLIGHT_IR691_2017-08-01_2017-09-01.csv.bz2

139. Also note: where merge_apds_data_on_day takes two dates, they must be the start and end dates from the filename given to import_apds_file. merge_apds_data_on_day can also take four dates, where the first two dates are the date range and the last two dates are from the apds filename.

5.5 Monitoring Jobs

140. The jobs can be monitored from the GCP Kubernetes Engine Workloads page, see Figure 16.

141. Figure 16 shows a number of successfully launched jobs that are still running.

142. Whether a job launched successfully or not can be determined from the “status” column. Appendix A Troubleshooting describes some common errors and how they can be resolved.

143. Whether a job is still running or not can be determined from the “pods” column: “1/1” indicates that the job is still running, “0/1” indicates that the job has completed successfully.

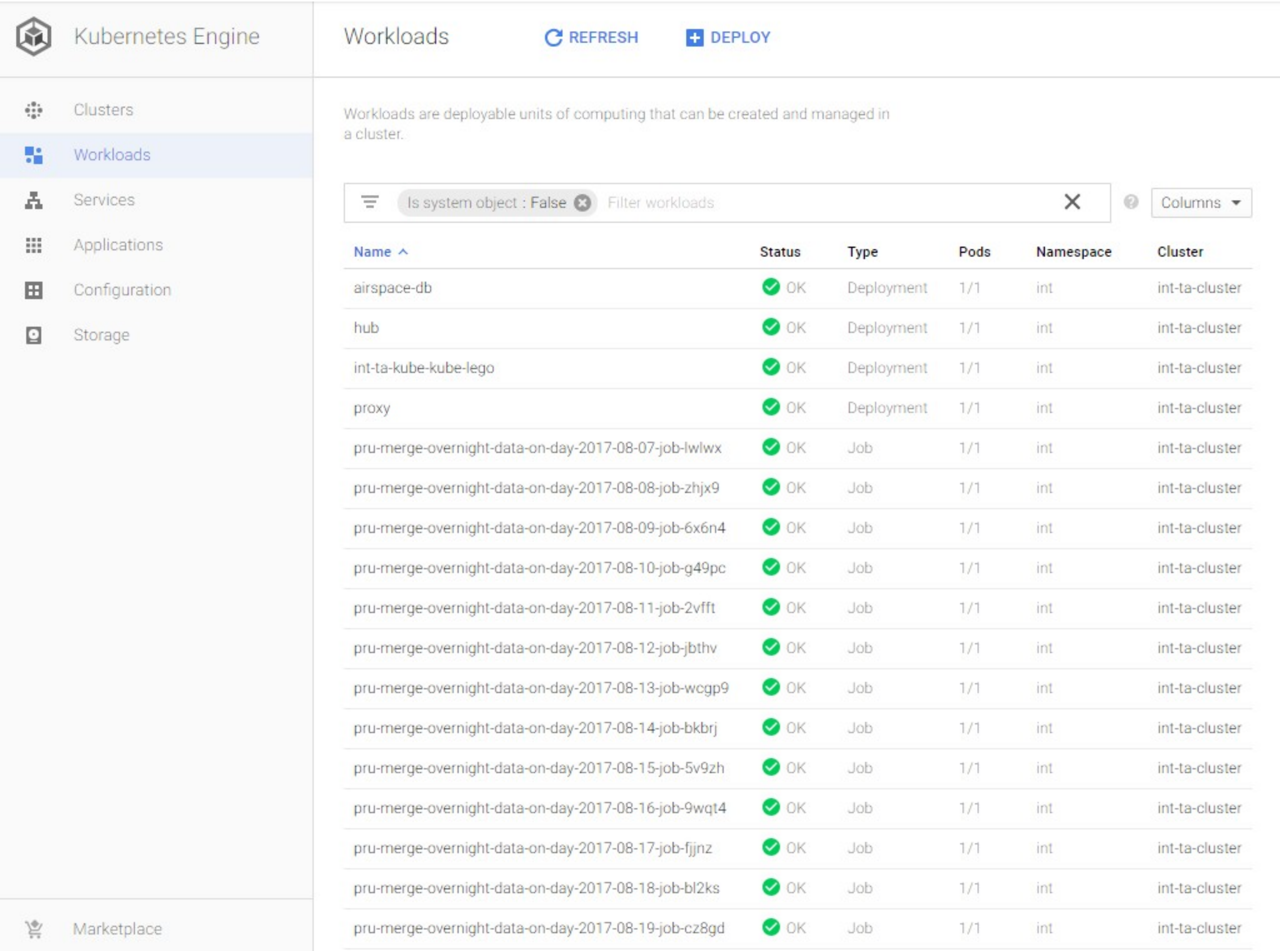

Figure 16: GCP Kubernetes Workloads page

144. The status of running jobs can also be found from the command line by running:

```
kubectl get jobs -n <namespace> -L proc,date
```

145. which outputs:


```
NAME                                          DESIRED   SUCCESSFUL   AGE       PROC
DATE
pru-import-data-on-day-2017-08-07-job-z8mqt   1         1            1h
import_data_on_day   2017-08-07
pru-import-data-on-day-2017-08-08-job-pzgvt   1         1            1h
import_data_on_day   2017-08-08
pru-import-data-on-day-2017-08-09-job-v8hqj   1         1            1h
import_data_on_day   2017-08-09
pru-import-data-on-day-2017-08-10-job-8vwpz   1         1            1h
import_data_on_day   2017-08-10
pru-import-data-on-day-2017-08-11-job-qprzt   1         1            1h
import_data_on_day   2017-08-11
pru-import-data-on-day-2017-08-12-job-gchvv   1         1            1h
import_data_on_day   2017-08-12
pru-import-data-on-day-2017-08-13-job-6kp7f   1         1            1h
import_data_on_day   2017-08-13
pru-import-data-on-day-2017-08-14-job-f6dcr   1         1            1h
import_data_on_day   2017-08-14
pru-import-data-on-day-2017-08-15-job-qnwxh   1         1            1h
import_data_on_day   2017-08-15
pru-import-data-on-day-2017-08-16-job-rmp8f   1         1            1h
import_data_on_day   2017-08-16
pru-import-data-on-day-2017-08-17-job-9xwpp   1         1            1h
import_data_on_day   2017-08-17
pru-import-data-on-day-2017-08-18-job-vcv8h   1         1            1h
import_data_on_day   2017-08-18
pru-import-data-on-day-2017-08-19-job-hgvtj   1         1            1h
import_data_on_day   2017-08-19
pru-import-data-on-day-2017-08-20-job-9nmdj   1         1            1h
import_data_on_day   2017-08-20
pru-import-data-on-day-2017-08-21-job-qvrcg   1         1            1h
import_data_on_day   2017-08-21
pru-import-data-on-day-2017-08-22-job-2jgrm   1         1            1h
import_data_on_day   2017-08-22
pru-import-data-on-day-2017-08-23-job-jwwk5   1         0            1h
import_data_on_day   2017-08-23
pru-import-data-on-day-2017-08-24-job-7z992   1         0            1h
import_data_on_day   2017-08-24
pru-import-data-on-day-2017-08-25-job-p629k   1         1            1h
import_data_on_day   2017-08-25
pru-import-data-on-day-2017-08-26-job-r4pf4   1         1            1h
import_data_on_day   2017-08-26
pru-import-data-on-day-2017-08-27-job-sr6g4   1         1            1h
import_data_on_day   2017-08-27
pru-import-data-on-day-2017-08-28-job-d6h64   1         1            1h
import_data_on_day   2017-08-28
pru-import-data-on-day-2017-08-29-job-nsdtd   1         1            1h
import_data_on_day   2017-08-29
pru-import-data-on-day-2017-08-30-job-vfq9g   1         1            1h
import_data_on_day   2017-08-30
pru-import-data-on-day-2017-08-31-job-dh8z8   1         0            1h
import_data_on_day   2017-08-31
```

146. Where a completed job has a 1 in the “SUCCESSFUL” column.

5.6 Reading Logs

147. Kubernetes logs the console output of all pods and jobs.

148. The log for a job can be read by selecting the job name on the GCP Kubernetes Engine Workloads page (see Figure 16).

149. This brings up the Job details page for the selected job, see Figure 17.

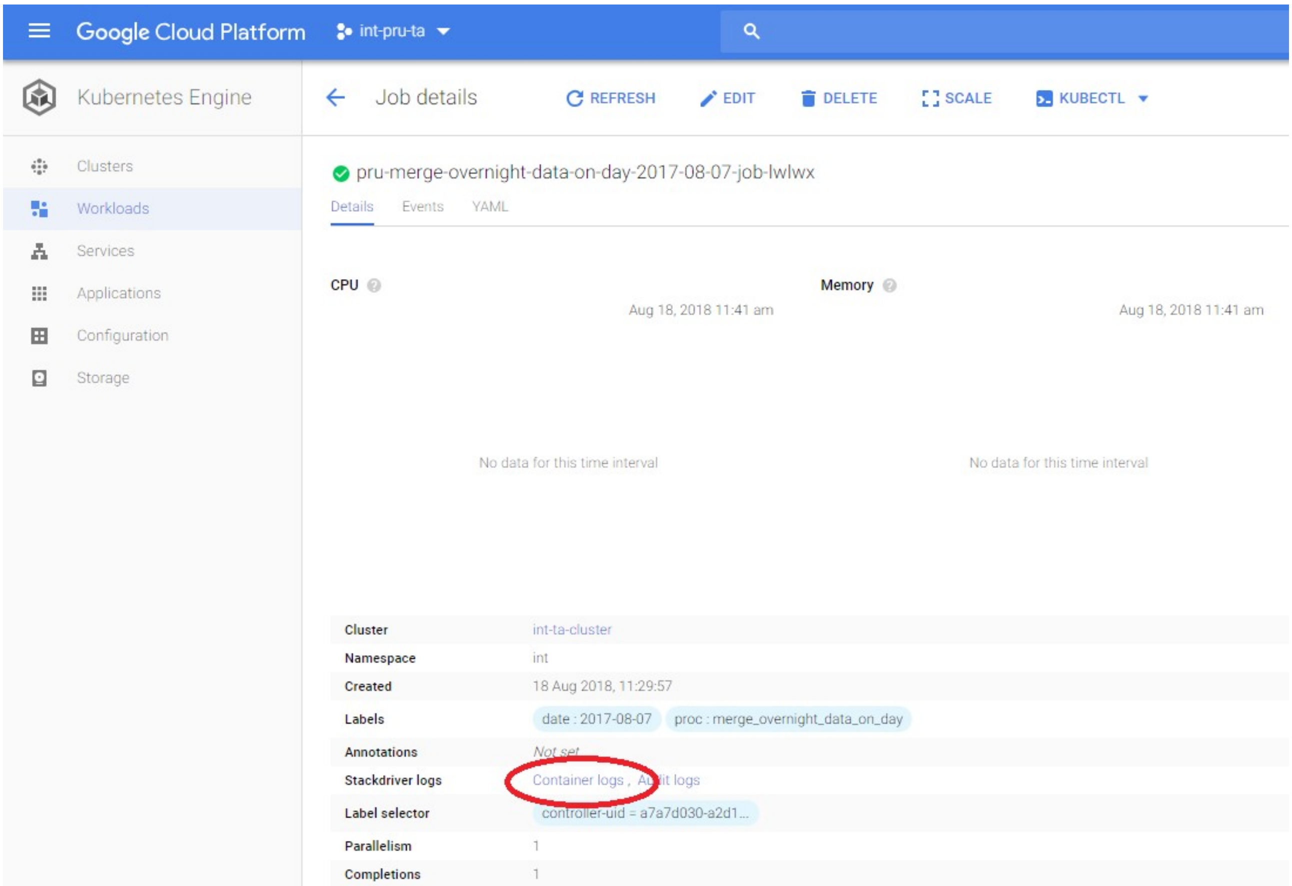

Figure 17: GCP Kubernetes Engine Job details page

150. Clicking on “Container logs” on the Job details page (see red ellipse in Figure 17) brings up the GCP Stackdriver Logging page, see Figure 18.

151. The log usually opens at the most recent entry. The down arrow by the “Jump to now” button can be used to “Jump to first entry” and the scroll bar can be used to scroll through log entries.

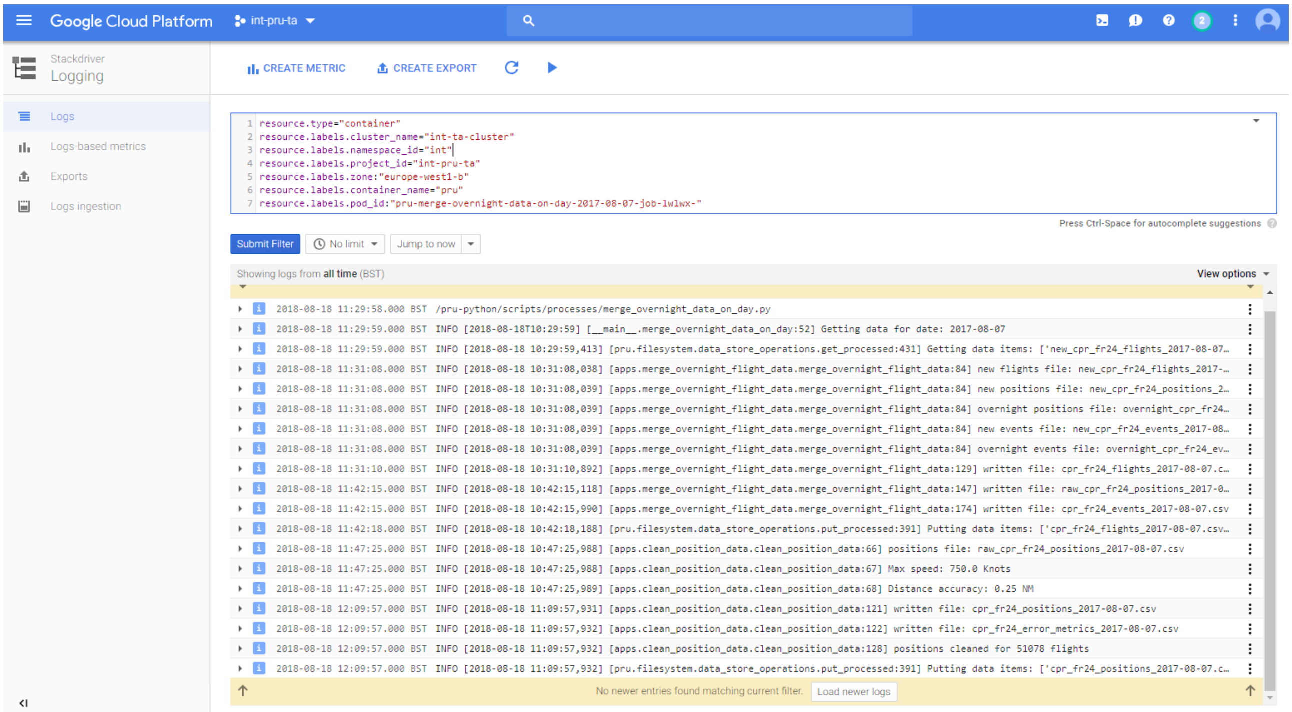

Figure 18: GCP Stackdriver log page

5.7 Killing Jobs

152. Kubernetes does not kill completed jobs. The jobs are left so that there logs can be inspected. The user must kill completed jobs manually.

5.7.1 Kill an individual job

153. An individual job can be killed with the following command:

```
kubectl delete jobs -n <namespace> <job name>
```

154. For example:

```
kubectl delete jobs -n int pru-import-data-on-day-2017-08-07-job-z8mqt
```

5.7.2 Kill all jobs

155. All jobs can be killed with the following command:

```
kubectl delete jobs -n <namespace> --all
```

156. For example:

```
kubectl delete jobs -n int --all
```


5.7.3 Kill jobs running a process

157. All the jobs running a process can be killed with:

```
kubectl delete jobs -n <namespace> -l proc=<process name>
```

158. For example:

```
kubectl delete jobs -n int -l proc=import_data_on_day
```

5.8 Disabling Compute Resources

159. GCP charges for all the compute nodes enabled in a Kubernetes cluster regardless of whether they are used or not. Therefore, it is recommended to disable unused compute nodes, by editing the cluster configuration, setting their node pool sizes to zero, see Figure 15

160. Note: GCP takes around a minute to shutdown each node in a node pool, e.g. it may take over 30 minutes to shutdown low memory process nodes for a months data if they use individual compute nodes for each kubernetes job.

Appendix

A

A Troubleshooting

161. The following are some common errors encountered while running jobs on kubernetes and what you can do to resolve them.

A.1 Unscheduleable

162. This error is encountered when there are not enough resources for kubernetes to schedule a job.

163. Clicking on the word “unschedulable” should tell you why the job could not be scheduled, i.e. “insufficient cpu” or “insufficient memory”, or both.

A.1.1 Solution

164. First wait a minute to ensure that the job really is unschedulable. Often kubernetes simply needs more time to schedule a job.

165. The solution is to provision more compute resources or kill the job and run it again after other jobs have finished.

A.2 OOMKilled

166. This error is encountered when a running job exceeds its kubernetes memory resource limit.

A.2.1 Solution

167. The job needs more memory to run, consider increasing kubernetes memory requests/and or limits for the job.

A.3 CrashLoopBackoff

168. This error is encountered when there is an issue with the process running in the container. For example a PostGIS database not being available or a bug the pipeline process.

A.3.1 Solution

169. Collect logs from the job and contact Via Technology.

