Add End Time To Asynchronous Jobs

Introduction

This enhancement enables Cloudstack users to establish how long an asynchronous job took to complete. By recording the start time and end time of an asynchronous job, users will able to extract statistical data regarding the performance of various types of asynchronous jobs. This data can then be used for reporting.

Feature Specification

CloudStack has the concept of executing certain API calls asynchronously when they take a long period of time to complete. They will immediately return a job id of the job that will be responsible for executing the command. This job id can be used to query the status of the job by using the queryAsyncJobResult application programming interface (API) call.
Among other fields, the result of this API query returns the 'created' field, which is the timestamp of when an asynchronous job started. Before this feature, there was no functional mechanism that captured and persisted the end time of when an asynchronous job has finished. As a result, the "queryAsyncJobResult" API did not return a 'completed' field.

API

- Cloudstack now captures the job end timestamp of when the asynchronous job has finished and populates this into the existing database field called 'removed' in the async_job table.
- Please note the 'removed' field is not currently being used anywhere else in the CloudStack code, and the 'removed' database column is also not populated by any other processes.
- A new response tag is now added to the queryAsyncJobResult API called 'completed'.
- When making a queryAsyncJobResult API request, the value of the database column 'removed' is now mapped to this 'completed' response tag.
- The queryAsyncJobResult API field is called 'completed' instead of 'removed' because it will be more descriptive to an API user.

Management Server Process

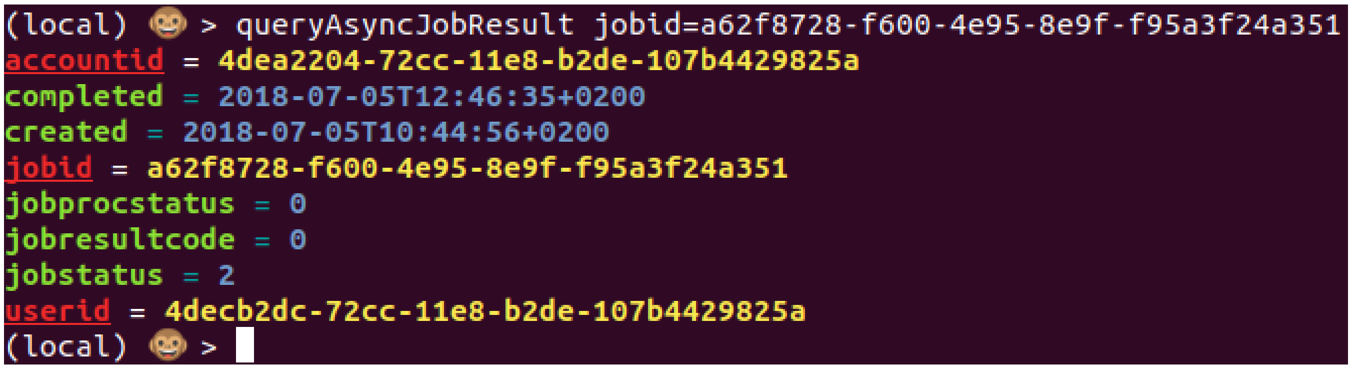
- When an asynchronous job completes it is marked as complete by updating its finished status in the database.
- New functionality is added to also update the 'removed' field with the timestamp of when an asynchronous job has completed.
- In addition, when the Cloudstack management server is stopped and started again (gracefully or ungracefully), neither this management server nor other running management servers have any knowledge of the true status of the asynchronous jobs that completed during this time it was down.
- Their statuses are not tracked or updated in the database by any management server during this time, regardless of whether they are still running, finished successfully or finished with an error status.
- Currently, there is no mechanism to notify other running management servers that a specific management server is stopping or to notify a specific running management server that it should start to monitor/track the currently running asynchronous jobs belonging to the management server being stopped.
- When a Cloudstack management server starts up, it does not do a blanket delete of the asynchronous jobs it is the owner of.
- Instead, it finds in the database all the asynchronous jobs it is the owner of, whose statuses are in an 'in_progress' state and updates them to a 'failed' status.
- At the same time, it now also marks them as 'removed' by updating the 'removed' field with the current timestamp.

Garbage Collection

- CloudStack has a garbage collector that does database clean-up of asynchronous jobs, whereby it periodically deletes unfinished and completed job records from the async_job database table.
- It uses the configurable global setting 'job.cancel.threshold.minutes' to cancel jobs that are still in the queue.
- It also uses a configurable global setting 'job.expire.minutes' that allows a user to specify how long in minutes to keep asynchronous jobs that have not been processed yet, as well as completed asynchronous job records in the database before deleting them.
- Unfinished jobs that haven't been processed yet and that are older than this expiry time will be expired and deleted by the garbage collector.
- Asynchronous jobs that have completed before the timeout threshold will also be deleted.
- When these jobs complete they are marked as complete by updating their finished status in the database.
- The garbage collector used to use the 'created' database column to find completed asynchronous job records that are older than the 'job.expire.minutes' global setting's timestamp.
- It now uses the 'removed' column to find these completed asynchronous job records instead.
- Due to the nature of the garbage collector, any reporting that needs to be done on asynchronous jobs, should be done before the garbage collector starts its cleaning up task.

Screenshots

A call to queryAsyncJobResult API returns both the created and the endtime field values.



```
(local) 🐵 > queryAsyncJobResult jobid=a62f8728-f600-4e95-8e9f-f95a3f24a351
accountid = 4dea2204-72cc-11e8-b2de-107b4429825a
completed = 2018-07-05T12:46:35+0200
created = 2018-07-05T10:44:56+0200
jobid = a62f8728-f600-4e95-8e9f-f95a3f24a351
jobprocstatus = 0
jobresultcode = 0
jobstatus = 2
userid = 4decb2dc-72cc-11e8-b2de-107b4429825a
(local) 🐵 >
```