### Arvados - Feature #13773

"Will fail" status for failing (but not yet failed) containers

07/09/2018 04:55 PM - Peter Amstutz

<table>
<thead>
<tr>
<th>Status:</th>
<th>Resolved</th>
<th>Start date:</th>
<th>09/06/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Assigned To:</td>
<td>Lucas Di Pentima</td>
<td>% Done:</td>
<td>100%</td>
</tr>
<tr>
<td>Category:</td>
<td></td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td>Target version:</td>
<td>2018-09-19 Sprint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

It is useful to know when a container is going to fail, but hasn't completed yet.

Workflow developers / users want to know this so workflows can be resubmitted.

arvados-cwl-runner wants to use this to avoid reusing an arvados-cwl-runner container which has already decided to fail.

**Proposed implementation**

**API**

Add a runtime_status serialized hash attribute to containers model on API server, stored as an indexed jsonb column.

- runtime_status can be updated when state∈{"Locked", "Running"}.
- runtime_status is cleared if state changes from "Locked" to "Queued" (to avoid leaking status messages between different dispatch attempts).

If a container with state="Running" has an error key in its runtime_status then it must not be a candidate for reuse. (Note "Locked" state is deliberately omitted here because dispatch/setup errors are retryable.)

**Documentation**

Well known keys in runtime_status should be documented on the container schema page:

- error: string, indicates the container will definitely fail, or has already failed
- warning: string, indicates something unusual happened or is currently happening, but isn't considered fatal
- activity: string, a message for the end user about what state the container is currently in

**arvados-cwl-runner**

- store the first fatal error (failed child, error in workflow definition) in error
- (secondary goal) mention any additional errors ("first error (4 additional errors)?")
- (secondary goal) store jobsComplete / jobsWaiting / jobsFailed

**Workbench**

If a running container has error or warning in its runtime_status, Workbench should flag it with a color/label to distinguish it from the normal "running" state (perhaps also showing the error/warning message in a tooltip) on the dashboard and other summary views.

Workbench should display any error or warning messages prominently in the detailed view.

**Additional ideas**

These features are anticipated but they are not expected to be included in the initial implementation:

- crunch-dispatch-slurm can update the activity field to indicate "in slurm queue"
- crunch-run can update the activity field to indicate loading Docker image or uploading output
- crunch-run or arv-mount can detect likely cache thrashing conditions and generate a warning
- arvados-cwl-runner reports additional structured error details under errorDetails for Workbench to display

**Subtasks:**

- Task # 13843: Review 13773-will-fail-container-status Resolved
Related issues:
Related to Arvados - Bug #13772: Rerunning a container_request that has a fail...

New

Associated revisions
Revision 1010c9e5 - 09/13/2018 08:46 PM - Lucas Di Pentima
Merge branch '13773-will-fail-container-status'
Closes #13773
Arvados-DCO-1.1-Signed-off-by: Lucas Di Pentima <ldipentima@veritasgenetics.com>

History
#1 - 07/09/2018 04:55 PM - Peter Amstutz
- Status changed from New to In Progress

#2 - 07/09/2018 05:00 PM - Peter Amstutz
- Subject changed from "Will fail" status to prevent reuse of failing (but not yet failed) container to "Will fail" status for failing (but not yet failed) containers
- Description updated
- Status changed from In Progress to New

#3 - 07/09/2018 05:02 PM - Peter Amstutz
- Description updated

#4 - 07/09/2018 07:13 PM - Tom Clegg
- Description updated

#5 - 07/11/2018 02:18 PM - Peter Amstutz
Introduce a "status" field with some well defined fields.

#6 - 07/11/2018 03:45 PM - Peter Amstutz
- Description updated

#7 - 07/11/2018 05:12 PM - Peter Amstutz
- Description updated

#8 - 07/11/2018 05:14 PM - Tom Morris
- Target version changed from To Be Groomed to Arvados Future Sprints
- Story points set to 3.0

#9 - 07/18/2018 03:40 PM - Tom Morris
- Target version changed from Arvados Future Sprints to 2018-08-01 Sprint

#10 - 07/18/2018 03:40 PM - Lucas Di Pentima
- Assigned To set to Lucas Di Pentima

#11 - 07/18/2018 04:12 PM - Tom Clegg
- Description updated

#12 - 07/18/2018 04:56 PM - Peter Amstutz
The idea of having error/warning be a list was to help communicate multiple errors (eg steps 2, 5 and 7 failed).
It should be possible to update runtime_status when in "Locked" state, because currently the locked state includes the container initialization (we'd like to communicate things like "current activity is loading Docker image".

#13 - 07/19/2018 01:56 PM - Tom Clegg
- Description updated
Regarding multiple errors: the "error" or "warning" values should be a single human-readable summary of the error/warning condition. Rather than giving an array of values and letting Workbench guess which ones are most important to display, clients should

- choose an appropriate way to report the multiple-error condition to a user, and store that string in the "error" field (e.g., "first error", "last error", "first error (N more errors)")
- optionally store an array of individual errors (and any other error details) under a different key like errorDetails.

Updated description to allow updates when state=Locked, and clear when state=Queued.

#14 - 07/19/2018 02:02 PM - Peter Amstutz
Tom Clegg wrote:

Regarding multiple errors: the "error" or "warning" values should be a single human-readable summary of the error/warning condition. Rather than giving an array of values and letting Workbench guess which ones are most important to display, clients should

- choose an appropriate way to report the multiple-error condition to a user, and store that string in the "error" field (e.g., "first error", "last error", "first error (N more errors)")
- optionally store an array of individual errors (and any other error details) under a different key like errorDetails.

So, my thought was that if it is an array of errors, workbench can just present it as a bulleted list. If it is a single string, and there are multiple errors, then the client could format them, but then we need to define how workbench is supposed to handle the formatting. Maybe allow textile in that field?

#15 - 07/19/2018 02:18 PM - Tom Clegg
I still think the "error" value, which all presentation code is expected to handle, is most useful as a plain old string. This makes it usable in a terminal window, a tooltip, a program that wants to wrap an error ("baz workflow failed: $x"), etc. The reporting client is welcome to store additional structured information in other keys -- but it still has to come up with a string, too. (If it's hard for arvados-cwl-runner to summarize its own multiple-error conditions usefully, I don't think it will get any easier in Workbench...)

#16 - 07/19/2018 02:31 PM - Peter Amstutz
Tom Clegg wrote:

I still think the "error" value, which all presentation code is expected to handle, is most useful as a plain old string. This makes it usable in a terminal window, a tooltip, a program that wants to wrap an error ("baz workflow failed: $x"), etc. The reporting client is welcome to store additional structured information in other keys -- but it still has to come up with a string, too. (If it's hard for arvados-cwl-runner to summarize its own multiple-error conditions usefully, I don't think it will get any easier in Workbench...)

So assume preformatted plain text (ie respect all line breaks)? That's fine, we just need to be clear about it. One of the goals here is for arvados-cwl-runner to be able to surface errors which came from a failing step instead of making the user sift through logs. However right now that means returning the last 40 lines of the step's stdout/stderr (because we don't know exactly where the error message is.)

#17 - 07/19/2018 02:39 PM - Peter Amstutz
Maybe think about what how we want errors to be presented on workbench, and then work back from there? Separate "error" (one line summary) and "error_details" (longer, formatted) would make sense.

#18 - 07/19/2018 03:50 PM - Tom Clegg
- Description updated

#19 - 07/20/2018 04:12 AM - Tom Clegg
- Description updated

#20 - 07/27/2018 03:26 PM - Lucas Di Pentima
- Status changed from New to In Progress

#21 - 08/01/2018 03:22 PM - Lucas Di Pentima
- Target version changed from 2018-08-01 Sprint to 2018-08-15 Sprint

#22 - 08/15/2018 03:11 PM - Lucas Di Pentima
- Target version changed from 2018-08-15 Sprint to 2018-09-05 Sprint

#23 - 09/05/2018 03:12 PM - Lucas Di Pentima
- Target version changed from 2018-09-05 Sprint to 2018-09-19 Sprint

#24 - 09/06/2018 03:07 PM - Lucas Di Pentima
Updates at 4a466cc5 - branch 13773-will-fail-container-status
Test run: https://ci.curoverse.com/job/developer-run-tests/876/

- Add runtime_status Hash attribute to containers on the API Server
- Avoid container reuse when there's an error key on runtime_status
- Add documentation about error, errorDetails, warning & activity keys
- On a-c-r, update runtime_status on runner containers when a child fails, including the child's name & uuid on the error key and the last 40 log lines on errorDetails.
- Add/update (and N more) string on the error key when more than 1 child fails.
- On workbench, show "Failing" and "Warning" labels on running containers with relevant runtime_status information. On the dashboard also show a tooltip with error/warning messages and on the detailed view show a panel, including errorDetails in the error case.
- Add tests on API Server & arvados-cwl-runner. Workbench tests pending.

#25 - 09/06/2018 07:31 PM - Peter Amstutz
If only certain errors get recorded, people are going to be confused and we'll be fielding support requests from users who don't know when to look in the logs tab. For example, the current implementation wouldn't show CWL syntax errors in runtime_status.

In arvados-cwl-runner, how hard would it be to intercept logging.exception(), logging.error(), and logging.warning() and have that call runtime_status_error()? I think this is mostly a matter of configuring the root logger with a custom output handler.

#26 - 09/06/2018 07:55 PM - Peter Amstutz
In workbench:

```
<div id="errorDetail" class="collapse">
  <pre>&lt;%= wu.runtime_status[:errorDetail] %&gt;</pre>
</div>
```

Does this sanitize :errorDetail (and other places where the :error message is rendered)?

Minor documentation nitpicking:

"error": "This container won't be successful because at least one step have already failed."

Should be "has already failed."

h2(#runtime_status). {% include 'container_runtime_status' %}

This doesn't need to be included from a separate file, the reason the other sections are that way is because they are included in both containers and container_requests, but this one is container specific.

#27 - 09/06/2018 09:44 PM - Peter Amstutz
Workbench presentation:

- I suggest using a similar layout to the workflow steps. This would be a panel where the title of the panel is the error message followed by a little down-pointing triangle. The body of the panel will be initially hidden, and clicking anywhere on the error message opens the panel body to display the full error.
- If there is a warning, it should always be displayed (not if/else with error).
- The value of "activity" isn't displayed on Workbench at all.

A couple additional points:

- The API server should validate that error, errorDetails, warning and activity are strings in runtime_status, if present.
- Should we add "warningDetails"?

Nitpicking aside, this looks really good, I tried it out with a test case (a workflow with one failing and one passing step) and it worked exactly as expected.

#28 - 09/10/2018 07:59 PM - Lucas Di Pentima

- File runtime status error warning.png added

I've written a log handler that intercepts log messages and updates runtime_status as an error/warning/activity status. When the log string contains a 'n' character, it uses it as a "kindDetail" field. The issue that I'm now having is on the error case: when a child container fails, the error log on the runner may arrive before the child running its runner's update status, so the errorDetail that we got from the 40 lines of error logs now are lost because only the first error occurrence is saved.

I've attached an example screenshot to also show how errors and warnings are now displayed simultaneously. I didn't make it work this way at first because I supposed that if there's an error, a warning is superfluous.

Regarding the activity status, do you have something in mind as which UI element to be used on the detailed/dashboard views?
My idea was that you’d take the first line of the first error message to use as "error", and then all subsequent error logging would get appended to errorDetail so that everything noteworthy is being displayed.

Warnings should handled the same way.

Activity should only be the most recent status. I don't think it makes sense for activity to be the same as the "info" log level, it should probably be better thought out as its own feature (and maybe incorporate updating the "progress" field as well.)

Tom has pointed out that detailed error reporting is out of scope for the ticket, so disregard my comments, let's merge the minimal version of this feature.

Intercept logging (warning & error) messages to save them as runtime status messages. This logging handler will populate the [error|warning]Detail field when the logged message is multi-line

Update documentation as suggested

Update a-c-r test so that it checks the logging handler does its task

Enhance runtime status error/warning display on workbench's container detailed page

Sanitize runtime status information when displaying it on workbench

Add runtime_status data type validation on the api server (w/test)

Fixed test that failed when running the entire sdk/cwl suite.

Nitpick, "levelno" is an integer, and (ERROR, WARNING, INFO) are something like (40, 30, 20) so I think you could use >= instead (in fact, there's a "CRITICAL" logging level which is levelno=50)

I tried running a workflow. I think there's a bug. I ran a test workflow and it never submitted a child workflow. The last update to "activity" does not correspond to the last INFO line of stderr. I'm suspicious that something in the new logging may have failed silently and left the workflow runner deadlocked.

I tried running a workflow. I think there's a bug. I ran a test workflow and it never submitted a child workflow. The last update to "activity" does not correspond to the last INFO line of stderr. I'm suspicious that something in the new logging may have failed silently and left the workflow runner deadlocked.

if record.levelno == logging.ERROR:
    kind = 'error'
elif record.levelno == logging.WARNING:
    kind = 'warning'
elif record.levelno == logging.INFO:
    kind = 'activity'

I tried running a workflow. I think there's a bug. I ran a test workflow and it never submitted a child workflow. The last update to "activity" does not correspond to the last INFO line of stderr. I'm suspicious that something in the new logging may have failed silently and left the workflow runner deadlocked.

if record.levelno == logging.ERROR:
    kind = 'error'
elif record.levelno == logging.WARNING:
    kind = 'warning'
elif record.levelno == logging.INFO:
    kind = 'activity'

I tried running a workflow. I think there's a bug. I ran a test workflow and it never submitted a child workflow. The last update to "activity" does not correspond to the last INFO line of stderr. I'm suspicious that something in the new logging may have failed silently and left the workflow runner deadlocked.

I tried running a workflow. I think there's a bug. I ran a test workflow and it never submitted a child workflow. The last update to "activity" does not correspond to the last INFO line of stderr. I'm suspicious that something in the new logging may have failed silently and left the workflow runner deadlocked.

I tried running a workflow. I think there's a bug. I ran a test workflow and it never submitted a child workflow. The last update to "activity" does not correspond to the last INFO line of stderr. I'm suspicious that something in the new logging may have failed silently and left the workflow runner deadlocked.
#35 - 09/13/2018 07:32 PM - Lucas Di Pentima
Update at 6451f2e4a
Test run: https://ci.curoverse.com/job/developer-run-tests/892/

Updated logging level comparison as requested by Peter. Both ERROR & CRITICAL logging will go to error runtime_status.

#36 - 09/13/2018 07:36 PM - Peter Amstutz
Lucas Di Pentima wrote:

Update at 6451f2e4a
Test run: https://ci.curoverse.com/job/developer-run-tests/892/

Updated logging level comparison as requested by Peter. Both ERROR & CRITICAL logging will go to error runtime_status.

Thanks.

I was not on the latest branch when I ran the test workflow. On the latest branch, it worked as expected.

This LGTM.

#37 - 09/13/2018 08:08 PM - Lucas Di Pentima
Merged master into the branch, resolved a migration conflict at 30d37841e
Running tests one last time before merging: https://ci.curoverse.com/job/developer-run-tests/893/

#38 - 09/14/2018 12:19 AM - Lucas Di Pentima
- Status changed from In Progress to Resolved

Applied in changeset arvadosj1010c9e51940f0eb36971df6c382b7cc0e2732.

#39 - 11/13/2018 09:00 PM - Tom Morris
- Release set to 14

#40 - 04/01/2019 02:34 PM - Tom Clegg
- Related to Bug #13772: Rerunning a container_request that has a failed child CR should restart the failed CR added

Files
<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtime status error warning.png</td>
<td>78.6 KB</td>
<td>09/10/2018</td>
<td>Lucas Di Pentima</td>
</tr>
<tr>
<td>fail.cwl</td>
<td>537 Bytes</td>
<td>09/13/2018</td>
<td>Peter Amstutz</td>
</tr>
</tbody>
</table>