Arvados - Feature #9272

[Crunch2] Pass container auth token (not dispatch token) to arv-mount, and (if requested) into the container itself

05/24/2016 06:29 PM - Peter Amstutz

Status:	Resolved	**Start date:**	05/26/2016
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2016-06-08 sprint		

Description

Add crunch-run support for a mount which indicates that Arvados credentials should be provided in the environment.

- In crunch-run, use the new "auth" API (added in #8128) to retrieve the appropriate token.
- Pass it to the container as an ARVADOS_API_TOKEN environment variable if requested via mount.
- Pass it to arv-mount, instead of using the dispatch token like the current code does.

Subtasks:

Task # 9299: Retrieve & use correct token	**Resolved**
Task # 9291: Review 9272-use-container-auth	**Resolved**
Task # 9305: Review 9272-test-races	**Resolved**
Task # 9306: (extras) fix state transitions to match API rules	**Resolved**

Associated revisions

Revision bde564d8 - 05/27/2016 03:35 PM - Tom Clegg

Merge branch '9272-test-races'

refs #9272

Revision ebb21c7d - 06/03/2016 07:54 PM - Tom Clegg

Merge branch '9272-use-container-auth'

closes #9272

History

#1 - 05/24/2016 06:30 PM - Peter Amstutz

- Description updated

#2 - 05/24/2016 06:31 PM - Brett Smith

- Target version set to 2016-06-08 sprint

#3 - 05/24/2016 06:37 PM - Tom Clegg

- Subject changed from [Crunch2] Flag to indicate that container should get Arvados credentials with --env to [Crunch2] Pass container auth token (not dispatch token) to arv-mount, and (if requested) into the container itself

- Description updated

#4 - 05/25/2016 07:14 PM - Tom Clegg

- Assigned To set to Tom Clegg

#5 - 05/25/2016 07:14 PM - Tom Clegg

- Story points set to 1.0

#6 - 05/26/2016 04:00 PM - Tom Clegg

- Status changed from New to In Progress

#7 - 05/27/2016 01:40 AM - Tom Clegg

On 9272-use-container-auth,

- the first two commits address the story as described
- the next commit addresses the state transitions discussed in #9187 and IRC:
  - don't start the container before changing state to Running
  - don't set state to Complete if there's no output / exit code -- use Cancelled if the container is "lost"
  - reset state to Queued if there was a problem setting up the container (before changing state to Running)
- the next two commits are misc tiny improvements to the test suite

Probably best to deal with the first two commits first (i.e., review beae579) to make sure the present issue gets addressed, and then deal with the extras.

#8 - 06/02/2016 02:19 PM - Peter Amstutz

- It should call UpdateContainerRecordRunning() after StartContainer(), otherwise if StartContainer() fails it will try to go Running -> Queued
- Suggest introducing a RuntimeConstraints struct with "API" as *bool field instead of using map[string]interface{}.
- As we discussed yesterday, to solve the race described in #9328 the auth token needs to be passed in at the time the crunch-run process is dispatched. While that means code in this branch will have to change, I don't think we should block the merge on that story.

#9 - 06/02/2016 04:03 PM - Tom Clegg

Peter Amstutz wrote:

> - It should call UpdateContainerRecordRunning() after StartContainer(), otherwise if StartContainer() fails it will try to go Running -> Queued

I don't think we can call StartContainer without first setting state to Running: StartContainer might succeed, resulting in the container actually running, but the container record still marked Locked, and thus able to go back to Queued state. See Containers API ("container states").

AFAICT, with the docker library, this is the latest point where we can change state to Running (this is why I split "create and start" into two separate functions).

> - Suggest introducing a RuntimeConstraints struct with "API" as *bool field instead of using map[string]interface{}.

Yeah, that would be better. Now that #9162 is merged, perhaps I should add Container to the new SDK instead of having every program come up with its own version of it...

> - As we discussed yesterday, to solve the race described in #9328 the auth token needs to be passed in at the time the crunch-run process is dispatched. While that means code in this branch will have to change, I don't think we should block the merge on that story.

Agreed

#10 - 06/02/2016 04:26 PM - Peter Amstutz

Tom Clegg wrote:

> Peter Amstutz wrote:
>
> > - It should call UpdateContainerRecordRunning() after StartContainer(), otherwise if StartContainer() fails it will try to go Running -> Queued
>
> I don't think we can call StartContainer without first setting state to Running: StartContainer might succeed, resulting in the container actually running, but the container record still marked Locked, and thus able to go back to Queued state. See Containers API ("container states").

If we change it to Running before the container actually starts, then if Start() fails the existing code will go Running -> Complete which is clearly wrong. It should go Running -> Cancelled. The quick fix is to add runner.finalState = "Cancelled" to the error block right after StartContainer(). However this behavior would be inconsistent with the policy in crunch-run leading up to this point which is to to go back to Queued state if we are unable to start the container for any reason.

Once we do #9328, it is fine if crunch-run crashes between successfully starting the container and setting the container to Running. It should get cleaned up by the dispatcher, put Locked -> Queued, and can be started again. The token(s) from the old crunch-run would be revoked so the process is orphaned and can't cause any trouble.

#11 - 06/02/2016 04:33 PM - Peter Amstutz

Tom Clegg wrote:

> AFAICT, with the docker library, this is the latest point where we can change state to Running (this is why I split "create and start" into two separate functions).

Just to clarify case you were assuming otherwise, unlike "docker run", Docker.StartContainer() is not a blocking call. Docker.Wait() is where we block and wait for the container to terminate. This why I think it is better to set the Running state after calling StartContainer().

#12 - 06/02/2016 04:42 PM - Tom Clegg

Peter Amstutz wrote:

> Tom Clegg wrote:
>
> > AFAICT, with the docker library, this is the latest point where we can change state to Running (this is why I split "create and start" into two separate functions).
>
> Just to clarify case you were assuming otherwise, unlike "docker run", Docker.StartContainer() is not a blocking call. Docker.Wait() is where we block and wait for the container to terminate. This why I think it is better to set the Running state after calling StartContainer().

Thanks for clarifying. However, we need to change state to Running before the container *starts* (not merely before it terminates).

My understanding is that once we call StartContainer, the container might start at any time -- e.g., between the time StartContainer returns and the time a subsequent "update state" API call takes effect. Starting the container while the container record state is still Locked would violate the Containers API.

#13 - 06/02/2016 05:11 PM - Peter Amstutz

Tom Clegg wrote:

> Thanks for clarifying. However, we need to change state to Running before the container *starts* (not merely before it terminates).
>
> My understanding is that once we call StartContainer, the container might start at any time -- e.g., between the time StartContainer returns and the time a subsequent "update state" API call takes effect. Starting the container while the container record state is still Locked would violate the Containers API.

Ok. My comment about fixing it so it goes into "Cancelled" instead of "Complete" still holds. One failure mode I can think of would be if the program to be run turns out to not exist inside the container, which should definitely be a hard failure rather than being put back into the Queue.

#14 - 06/02/2016 05:38 PM - Tom Clegg

Peter Amstutz wrote:

> If we change it to Running before the container actually starts, then if Start() fails the existing code will go Running -> Complete which is clearly wrong. It should go Running -> Cancelled. The quick fix is to add runner.finalState = "Cancelled" to the error block right after StartContainer().

Yes, you're right. Fixed in 4a2495c.

> However this behavior would be inconsistent with the policy in crunch-run leading up to this point which is to to go back to Queued state if we are unable to start the container for any reason.

There is no atomic "start container and change API record" operation so we have to choose one of the following evils:

1. It is possible to go Locked→Running even though the user code never actually gets invoked
2. It is possible to start a container running user code, without ever changing Locked→Running

The API specifies that the first evil is the lesser one.

> Once we do #9328, it is fine if crunch-run crashes between successfully starting the container and setting the container to Running. It should get cleaned up by the dispatcher, put Locked -> Queued, and can be started again. The token(s) from the old crunch-run would be revoked so the process is orphaned and can't cause any trouble.

Not according to the API. Once a container has started (or "maybe started, we don't know"), it can't un-start. After this point, if we want to retry, we have to create a new container.

#15 - 06/02/2016 05:40 PM - Tom Clegg

Tom Clegg wrote:

> Peter Amstutz wrote:
>
> > - Suggest introducing a RuntimeConstraints struct with "API" as *bool field instead of using map[string]interface{}.

> Yeah, that would be better. Now that #9162 is merged, perhaps I should add Container to the new SDK instead of having every program come up with its own version of it...

Meh, decided to punt for now. Just changed to struct with *bool. 0d8b6bf

#16 - 06/03/2016 02:25 PM - Tom Clegg

Tom Clegg wrote:

> The API specifies that the first evil is the lesser one.

Sorry, this sounds weird. I should say "the API **currently** specifies". The fact that I wrote it on the API wiki doesn't mean it's the right and final answer. We can certainly change it if we decide the other evil is lesser.

#17 - 06/03/2016 03:07 PM - Peter Amstutz

LGTM @ 0d8b6bf

#18 - 06/03/2016 08:25 PM - Tom Clegg

- Status changed from In Progress to Resolved

- % Done changed from 75 to 100

Applied in changeset arvados|commit:ebb21c7daa50d4101b34647b0e961e4f470a5b0b.