**Description**

When creating a container request, a client can specify an API token to be used in a container. When running a container on a cluster other than the requesting user's home cluster, this allows the container to read collections on other clusters.

**runtime_token:**
- if provided, must be a valid v2 API token
- is not provided in API responses (similar to secret_mounts)
- can be retrieved by the dispatcher that has the lock using the /arvados/v1/containers/$uuid/auth API (as usual)
- can be set to a valid token when creating a container request
- is validated by the API server before creating the container request
- is provided instead of a real api_client_authorization record by .../$uuid/auth endpoint
- causes Container#assign_auth to be a no-op
- provide the cached remote token from .../$uuid/auth endpoint
- Container#assign_auth assigns the cached remote token
- is ignored when considering containers for reuse
- is scrubbed when the container is final

The given token's user_uuid and scopes are stored in two additional new container fields, runtime_user_uuid (string) and runtime_auth_scopes (jsonb). These are considered significant in container reuse decisions.

If runtime_token is not provided:

- user_uuid and scopes ("all") are stored in the container record
- existing containers qualify for reuse if their user_uuid and scopes are equal, or both are null

**Not included here:**

To preserve the current level of locking safety (avoiding having two containers fighting over container record state after a slurm/network failure causes a container to be requeued), the auth token needs to change each time the container changes from Queued to Locked state. This should be done (e.g., by adding a field to the token that doesn't change its validity, and checking for equality when updating lock-protected container fields), but will not be addressed in the present issue.

**Subtasks:**

Task # 14286: Review 14260-runtime-token

**Related issues:**

Related to Arvados - Feature #14262: [Controller] Specify runtime_token when ... Resolved 10/24/2018

**Associated revisions**

Revision dd3172b9 - 10/18/2018 05:44 PM - Peter Amstutz
Merge branch '14260-runtime-token' refs #14260
Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

Revision 4cca4c6b - 10/19/2018 02:11 PM - Peter Amstutz
Merge branch '14260-expired-token-delete' refs #14260
Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

**History**

#1 - 09/26/2018 03:58 PM - Tom Clegg
I think we were trying to arrange it so that it wasn't necessary to create an api_client_authorization record for runtime_token. However, ApiClientAuthorization.validate() already creates and returns a record as a way of caching the remote token. So either it needs to stop doing that, or strike the "container.auth_uuid will be nil" part of the design.

My current approach is also inelegant, since auth_uuid is nil, it has to take the container uuid and reverse searches for a valid container request and then gets the token, so there's a possible race if one container request was cancelled and replaced by another one.

It implementation will be simpler if we assign Container.auth_uuid the uuid of the object returned from from ApiClientAuthorization.validate. One drawback is that in order to forget the token after we're done with it, we would have to delete the api_client_authorization record in addition to clearing the "runtime_token" field.

A third idea I just came up with would be to add a flag to ApiClientAuthorization.validate to return an unsaved object, which might simplify some of the code (but not solve the race condition).

I was imagining we would add the runtime_token field to the containers table, as well as the container_requests -- much like secret_mounts -- and bypass the usual token validation path in the case where a container is being updated using its own runtime_token. Assuming this doesn't make a mess of the middleware, it would be more efficient, and address the races, if I'm following correctly.

A third idea I just came up with would be to add a flag to ApiClientAuthorization.validate to return an unsaved object, which might simplify some of the code (but not solve the race condition).
This sounds right. If validate() takes a "don't cache" flag, we don't have anything to clean up -- we would typically only check it once, when the CR is created. After that we wouldn't need to use the usual validate() code path anyway because we'd just look at the runtime_token for the container being updated. I think this still leaves us caching the token if the container does other API calls (besides "update self"), though...?

#14 - 10/11/2018 01:06 PM - Peter Amstutz

I think this still leaves us caching the token if the container does other API calls (besides "update self"), though...?

I think that's the key point. The token is going to be used by the container to communicate with the local arvados-controller (for example, for arv-mount). Unless we intend to also change the remote token caching behavior, there's almost no benefit in trying to avoid caching the token during container creation, because it is going to be cached anyway as soon as it gets used.

So I'll go with the implementation strategy where the remote token is recorded in the api_client_authorizations table until the container request is finalized.

#15 - 10/11/2018 01:11 PM - Peter Amstutz
- Description updated

#16 - 10/11/2018 02:09 PM - Tom Clegg

...until the container is finalized

Should this be "until the token expires"?
- If the token was issued locally, it shouldn't be revoked merely because someone used it as a runtime_token
- If the token was issued remotely, the local cache will expire soon anyway
- Delete-on-finalize seems like an opportunity for rare races with other operations using the same cached token (although I haven't come up with a specific harmful example)

Maybe "delete expired entries from api_client_authorizations" (in lib/sweep_trashed_objects.rb?) would be easier, and would clean up other cruft too?

#17 - 10/11/2018 04:00 PM - Peter Amstutz

Tom Clegg wrote:

...until the container is finalized

Should this be "until the token expires"?
- If the token was issued locally, it shouldn't be revoked merely because someone used it as a runtime_token
- If the token was issued remotely, the local cache will expire soon anyway
- Delete-on-finalize seems like an opportunity for rare races with other operations using the same cached token (although I haven't come up with a specific harmful example)

I was thinking that the rule would be that runtime_token is automatically expired when the request is finalized. But leaving it alone would be less surprising.

Maybe "delete expired entries from api_client_authorizations" (in lib/sweep_trashed_objects.rb?) would be easier, and would clean up other cruft too?

Yes, that's a good idea (and long overdue), I will add that.

#18 - 10/12/2018 09:08 PM - Peter Amstutz


#19 - 10/15/2018 03:11 PM - Peter Amstutz


#20 - 10/15/2018 06:15 PM - Peter Amstutz

14260-runtime-token @ 14aa2b6a0f662604248a45ab6b4db3e92e49053a

https://ci.curoverse.com/job/developer-run-tests/934/
adds runtime_token to container requests
adds runtime_token, runtime_user_uuid, and runtime_auth_scopes to containers
now assigns runtime_user_uuid at the point a container is created for a container request, container token user is runtime_user_uuid (when runtime_token isn't in use)
make container auth return runtime_token if available
extends the v2 token format to add an optional 4th position, which is used for tokens given to containers, and has the container uuid
update crunch-run to assign extended v2 tokens to the container
clear runtime_token when container and container request are finalized
update tests
update documentation

#21 - 10/16/2018 05:40 PM - Lucas Di Pentima
Just to let you know, f2a3cbe fails on 3 secret mounts related tests. (In the meantime, I continue reading)

#22 - 10/17/2018 01:22 PM - Lucas Di Pentima

- On services/api/app/models/container.rb
  - Line 288: Does runtime_auth_scopes need to be sorted for comparison? (like environment, runtime_constraints, etc)
  - On Container.for_current_token(): wouldn't be cleaner to just return the first element of the where clause, as it's the only one being used, and will allow to simplify the code where it gets called.

#23 - 10/17/2018 03:18 PM - Peter Amstutz
- Status changed from New to In Progress
- Target version changed from 2018-10-17 sprint to 2018-10-31 sprint
- Story points changed from 3.0 to 1.0

#24 - 10/18/2018 04:24 PM - Peter Amstutz
Lucas Di Pentima wrote:

- On services/api/app/models/container.rb
  - Line 288: Does runtime_auth_scopes need to be sorted for comparison? (like environment, runtime_constraints, etc)

Good idea. Done.

- On Container.for_current_token(): wouldn't be cleaner to just return the first element of the where clause, as it's the only one being used, and will allow to simplify the code where it gets called.

The reason was ContainerRequest.get_requesting_container chains it with c.select([:uuid, :priority]).first

However, as far as I can tell this is just a micro-optimization to avoid loading columns we don't use, that probably isn't worth the complexity. Simplified it.

14260-runtime-token @ 559679a061470337d555a3de519a0e86ad4cdd2

#25 - 10/18/2018 05:43 PM - Lucas Di Pentima
This LGTM, please merge.

#26 - 10/23/2018 06:35 PM - Peter Amstutz
- Status changed from In Progress to Resolved

#27 - 11/13/2018 08:53 PM - Tom Morris
- Release set to 14