Arvados - Feature #13619
[Controller] Federated multi object retrieval

06/13/2018 03:17 PM - Tom Clegg

Status: Resolved  Start date: 09/28/2018
Priority: Normal  Due date:
Assigned To: Peter Amstutz  % Done: 100%
Category:  Estimated time: 0.00 hour
Target version: 2018-10-03 Sprint

Description
See Federation implementation roadmap

This is a defined subset of the "list" API that supports federation.

Requests for container requests (→ can generalize to all endpoints) with filters [["uuid", in, [...]]] will be recognized by
arvados-controller. Divide the list based on cluster ids and send the appropriate query to each cluster with a salted token. Collect
and return the results.

A query includes remote uuids must not specify limit, offset, order, or additional filters. If any are provided, it is an error.

If there is a network error contacting a cluster, return that error status in the response.

If fewer items were returned by a cluster than were expected, arvados-controller should request additional pages until it has
determined all available items are returned (some items requested may still not be returned if they don't exist or are not readable by
the user).

The number of items in the [[uuid, in, [...]]] request cannot be larger than the max page size for responses.

Subtasks:
Task # 14235: Review 13619-fed-object-list  Resolved

Related issues:
Related to Arvados - Feature #14197: [controller] Federated container requests  Resolved  09/21/2018

Associated revisions
Revision f4707d4a - 10/03/2018 01:23 PM - Peter Amstutz
Merge branch '13619-fed-object-list' closes #13619

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

History
#2 - 06/13/2018 05:03 PM - Tom Morris
- Target version set to To Be Groomed

#3 - 08/21/2018 07:42 PM - Peter Amstutz
- Subject changed from [Controller] Federated listing API to [Controller] Federated multi object retrieval
- Description updated

#4 - 08/21/2018 09:06 PM - Peter Amstutz
- Description updated
- Story points set to 3.0

#5 - 08/21/2018 09:06 PM - Peter Amstutz
- Target version changed from To Be Groomed to Arvados Future Sprints

#6 - 09/11/2018 09:07 PM - Peter Amstutz
- Related to Feature #14197: [controller] Federated container requests added
Mostly done, but on re-reading the description, there are some details with paging that I haven't addressed.

- Read in filters ["uuid", "in", [...] and ["uuid", ",", ",", ...] and split them up based on cluster prefix
- If more than one cluster prefix is listed, engages the multi-cluster query code
- Check if filter size exceeds max page size (requires new config.yml configuration parameter) (with test)
- Error if limit, offset, or order is provided (with tests)
- Fails the whole query on error (with tests)
- Loop and request missing uuids if previous query returned results and not all results have been returned
- Implemented generically, supports containers, container requests and workflows.
- Updated documentation

In docs, remove some extra "must" in the "list method must:" items (and I think that should be "request must", not "method must")?

Instead of url.ParseQuery(req.URL.RawQuery) and loadParamsFromForm, could we use stdlib (*http.Request).ParseForm()?

loadParamsFromJson appears to be unused

In docs, remove some extra "must" in the "list method must:" items (and I think that should be "request must", not "method must")?

In (*genericFederatedRequestHandler) handleMultiClusterQuery(), r.(string) == "uuid" crashes if r isn't a string; maybe selects should be []interface{} instead of [interface{]}?

In (*genericFederatedRequestHandler) handleMultiClusterQuery(), "for f, f1 := range filters" -- what's the "1" referring to in "f1"? Could it be called "filter"?

In (*genericFederatedRequestHandler) remoteQueryUUIDs() will not terminate in various cases (remote keeps returning the same uuids, or entries without a "uuid" key) -- could fix by stopping the loop when found doesn't grow, rather than when the remote returns nothing

ParallelRemoteRequests should be Concurrent... and I wonder if this should be a server-wide limit, or renamed to indicate it's a per-incoming-request limit?

In (*genericFederatedRequestHandler) handleMultiClusterQuery(), it looks like filters="uuid", "in" [1 would crash

Default req concurrency should be 4 if not given in config file, not 0 (deadlock?)

Default page size should be 1000 if not given in config file, not 0

In docs, remove some extra "must" in the "list method must:" items (and I think that should be "request must", not "method must")?

In docs, remove some extra "must" in the "list method must:" items (and I think that should be "request must", not "method must")?

In docs, remove some extra "must" in the "list method must:" items (and I think that should be "request must", not "method must")?
Fixed.

- (*multiClusterQueryResponseCollector*)collectResponse() does way too much type-casting. Should check response status first, then decode into an appropriate struct type. E.g., if an element of items doesn't decode to a map[string]interface{} then we might as well fail at the decoding stage instead of ignoring this later in remoteQueryUUIDs().

Fixed.

- (*genericFederatedRequestHandler*)remoteQueryUUIDs() will not terminate in various cases (remote keeps returning the same uuids, or entries without a "uuid" key) -- could fix by stopping the loop when found doesn't grow, rather than when the remote returns nothing.

Fixed.

- ParallelRemoteRequests should be Concurrent... and I wonder if this should be a server-wide limit, or renamed to indicate it's a per-incoming-request limit?

Renamed to FederatedRequestConcurrency

- Instead of url.ParseQuery(req.URL.RawQuery) and loadParamsFromForm, could we use stdlib (*http.Request).ParseForm()?

Done, the tricky bit is buffering the body for the cases where you are going to proxy the request and need to read the body a second time.

- loadParamsFromJson appears to be unused

In a previous branch I had it looking in the json payload for cluster_id, but on further consideration I'm fairly certain none of our SDKs put in there (it goes in the URL query portion) so I removed the unnecessary complexity.

- Prefer params.Get("foo") != "" over if len(params["foo"])==1 { ... params["foo"][][0] } unless something like "?foo" is supposed to trigger the non-empty case

Fixed.

- "if len(params["select"]) == 1" treats ?select=foo&select=bar as if no select were given at all -- intentional?

Does the API server do the right thing if you supply that? eg does rails turn that into a list?

- Prefer remoteParams.Set("_method", "GET") over remoteParams["_method"] = [string]"GET"
  - ...or when initializing, might as well use remoteParams := url.Values["_method": {"GET"}, "count": {"none"}]

Fixed.

- In (*genericFederatedRequestHandler*) handleMultiClusterQuery(), r.(string) == "uuid" crashes if r isn't a string; maybe selects should be [string] instead of [interface]?

Fixed.

- In (*genericFederatedRequestHandler*) handleMultiClusterQuery(), "for _, f1 := range filters" -- what's the "1" referring to in "f1"? Could it be called "filter"?

Fixed.

- In (*genericFederatedRequestHandler*) handleMultiClusterQuery(), it looks like filters="uuid":"in" f1 would crash

Fixed.

- In (*genericFederatedRequestHandler*) handleMultiClusterQuery(), should un-pyramid the if ok && lhs == "uuid" block -- just needs "if col, ok := f12.(string); !ok || col != "uuid" { return false }

Fixed.

- Default req concurrency should be 4 if not given in config file, not 0 (deadlock?)
Peter Amstutz wrote:

Tom Clegg wrote:

- ParallelRemoteRequests should be Concurrent... and I wonder if this should be a server-wide limit, or renamed to indicate it's a per-incoming-request limit?

Renamed to FederatedRequestConcurrency

I wonder if Federated → MultiCluster would help suggest that it applies to an individual request?

Can we move these to a Limits section like Cluster configuration? I'm not sure "Limits" is the perfect category name but putting them at the top level doesn't seem like a great way to start. RequestLimits?

- "if len(params["select"]) == 1" treats ?select=foo&select=bar as if no select were given at all -- intentional?

Does the API server do the right thing if you supply that? eg does rails turn that into a list?

It seems to take the last one. (url.Values)Get() takes the first.

The way it is is fine for now. But longer term, I think we should be parsing the incoming request using the same (hopefully stdlib-based) approach across the board, then build new outgoing request(s) if needed. That way we would be sending the same "select" param value we're basing our response assumptions on, for example.

Peter Amstutz wrote:

Tom Clegg wrote:

- Default req concurrency should be 4 if not given in config file, not 0 (deadlock?)

Fixed.

- Default page size should be 1000 if not given in config file, not 0

Fixed.

Are these exported on purpose? (If we're going to export defaults, I think it would be better to do it through the Cluster config object itself, not controller.Handler.)

func (h *Handler) FederatedRequestConcurrency() int
func (h *Handler) MaxItemsPerResponse() int

In doc/api/methods/containers.html.textile.liquid I don't think we should claim to support federated create/update -- even if we do forward the API calls, there isn't really a use case.

Tom Clegg wrote:

I wonder if Federated → MultiCluster would help suggest that it applies to an individual request?

Can we move these to a Limits section like Cluster configuration? I'm not sure "Limits" is the perfect category name but putting them at the top level doesn't seem like a great way to start. RequestLimits?
Done

Are these exported on purpose? (If we're going to export defaults, I think it would be better to do it through the Cluster config object itself, not controller.Handler.)

Added methods to arvados.Cluster.RequestLimits

In doc/api/methods/containers.html.textile.liquid I don't think we should claim to support federated create/update -- even if we do forward the API calls, there isn't really a use case.

Right, fixed.

13619-fed-object-list @ cd52bc7ca67cb315e9d6175df4c750ae190b9207e


#17 - 10/03/2018 01:22 PM - Tom Clegg
LGTM, thanks

#18 - 10/03/2018 03:23 PM - Peter Amstutz
- Status changed from In Progress to Resolved

#19 - 11/13/2018 08:49 PM - Tom Morris
- Release set to 14