Arvados - Story #14287

[Controller] Refactor API code / routing

10/03/2018 03:42 PM - Tom Clegg

Status:	Resolved	**Start date:**	06/18/2019
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2019-11-06 Sprint		

Description

Diagram:

https://docs.google.com/drawings/d/1Qsj7Re4kE1tNMy7RaNR_dc9U5t6vswMJOhCl2YlbsCM

Subtasks:

Task # 14289: Review lib/controller/router in 14287-controller-structure	**Resolved**
Task # 15374: Review lib/controller (minus router) in 14287-controller-structure	**Resolved**
Task # 15375: Review 14287-controller-structure (minus lib/controller)	**Resolved**
Task # 15463: Review 14287-federated-list	**Resolved**

Related issues:

Related to Arvados - Story #14001: [Spike] [Controller] Port "update workflow...	**New**	
Related to Arvados - Story #15922: Change EnableBetaController14287 (default ...	**Resolved**	**12/12/2019**

Associated revisions

Revision 9becfce0 - 05/21/2019 07:48 PM - Tom Clegg

14287: Merge branch 'master' into 14287-controller-structure

refs #14287
14287

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision 678f1f53 - 06/27/2019 05:34 PM - Tom Clegg

Merge branch '14287-controller-structure'

refs #14287

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision 8ead0691 - 07/23/2019 05:24 PM - Tom Clegg

Merge branch '14287-federated-list'

refs #14287

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision d7c47dc4 - 10/31/2019 07:01 PM - Tom Clegg

Merge branch '14287-federated-list'

fixes #14287

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

History

#1 - 10/03/2018 03:43 PM - Tom Clegg

- Story points set to 2.0

#2 - 10/03/2018 04:12 PM - Tom Clegg

- Target version changed from 2018-10-17 sprint to 2018-10-31 sprint

#3 - 10/17/2018 04:23 PM - Tom Clegg

- Target version changed from 2018-10-31 sprint to Arvados Future Sprints

#4 - 12/12/2018 04:57 PM - Tom Morris

- Target version changed from Arvados Future Sprints to To Be Groomed

#5 - 12/12/2018 05:00 PM - Tom Clegg

- Status changed from New to In Progress

- Target version changed from To Be Groomed to 2018-12-21 Sprint

#6 - 01/02/2019 04:13 PM - Tom Clegg

- Target version changed from 2018-12-21 Sprint to 2019-01-16 Sprint

#7 - 01/16/2019 04:14 PM - Tom Clegg

- Target version changed from 2019-01-16 Sprint to 2019-01-30 Sprint

#8 - 01/30/2019 04:12 PM - Tom Clegg

- Target version changed from 2019-01-30 Sprint to 2019-02-13 Sprint

#9 - 02/13/2019 04:04 PM - Tom Clegg

- Target version changed from 2019-02-13 Sprint to 2019-02-27 Sprint

#10 - 02/27/2019 04:22 PM - Tom Morris

- Target version changed from 2019-02-27 Sprint to 2019-03-13 Sprint

#11 - 03/01/2019 07:33 PM - Tom Morris

- Release set to 15

#12 - 03/13/2019 01:29 PM - Tom Clegg

- Target version changed from 2019-03-13 Sprint to 2019-03-27 Sprint

#13 - 03/27/2019 03:08 PM - Tom Clegg

- Target version changed from 2019-03-27 Sprint to 2019-04-10 Sprint

#14 - 03/27/2019 03:35 PM - Tom Clegg

- Related to Story #14001: [Spike] [Controller] Port "update workflow" API to controller added

#15 - 04/09/2019 06:13 PM - Tom Clegg

This is still a work in progress, but it's getting stable.

functional todos:

- fix x-request-id propagation
- salt tokens for federated requests

14287-controller-structure @ c1f8c92f0bc1b97e4dde0dc66c746b89490ac16d

#16 - 04/10/2019 03:08 PM - Tom Morris

- Release changed from 15 to 22

#17 - 04/10/2019 03:09 PM - Tom Clegg

- Target version changed from 2019-04-10 Sprint to 2019-04-24 Sprint

#18 - 04/11/2019 07:28 PM - Tom Clegg

14287-controller-structure @ d072b78d52285f13c2599a290b773b7c56d44b80

- rebased
- salted tokens
- x-request-id propagation

- new code passes the existing federation tests

#19 - 04/24/2019 03:30 PM - Tom Clegg

- Target version changed from 2019-04-24 Sprint to 2019-05-08 Sprint

#20 - 05/08/2019 05:22 PM - Tom Clegg

- Target version changed from 2019-05-08 Sprint to 2019-05-22 Sprint

#21 - 05/22/2019 05:03 PM - Tom Clegg

- Target version changed from 2019-05-22 Sprint to 2019-06-05 Sprint

#22 - 06/05/2019 03:00 PM - Tom Clegg

- Target version changed from 2019-06-05 Sprint to 2019-06-19 Sprint

#23 - 06/12/2019 01:28 PM - Tom Clegg

- Description updated

#24 - 06/19/2019 03:09 PM - Tom Clegg

- Target version changed from 2019-06-19 Sprint to 2019-07-03 Sprint

#25 - 06/19/2019 07:53 PM - Lucas Di Pentima

Reviewing 14287-controller-structure branch (minus lib/controller/*): LGTM. Just wondering if it's really necessary to include the Specimen APIs, its that needed to support preexisting tests?

#26 - 06/19/2019 08:42 PM - Peter Amstutz

router/checker.go

I started going alphabetically and opened this file first and was confused as to what it was for (I guess it is used in testing). Could use a comment at the top.

router/error.go

There's no standard struct for errorWithStatus ? This looks very familiar (but maybe I'm just thinking of earlier iterations of controller.)

router/router.go

loadRequestParams() I don't see handling for application/x-www-form-urlencoded but I think that's because req.Form already parses it for us. Could you add a comment?

```
                // Delete field(s) that appear in responses
                // but not in update attrs, so clients can
                // fetch-modify-update.
                delete(v, "etag")
                delete(v, "unsigned_manifest_text")
```

There's a number of read-only fields like this, why filter these two in particular? (For example, at one point workbench2 was doing read-modify-update and sending back collection_version which resulted in an error.)

```
                rtr.mux.HandlerFunc(method, "/"+route.endpoint.Path, func(w http.ResponseWriter, req *http.Request
) {
                                        ...
                })
```

Instead of a large, deeply nested anonymous function, it would probably be more maintainable to put the body in its own function.

Instead of transcode() doing a full serialize/deserialize could we use something like mapstruct instead? (The actual goal seems to be to copy fields between struct and map[string]interface{}). This JSON serialize approach likely creates multiple copies of keys and values (which could be a concern for large fields like mounts or manifest_text).

router/response.go

sendResponse()

Is the ResponseWriter primed to send the "Content-Type: application/json" header? I don't see it (or any other headers) being set.

This seems to have three or four strategies to infer what the 'kind' field should be, which feels hacky. Is there a better way to propagate/preserve that information from whatever generated the response originally? If we're having round-tripping problems propagating API responses then maybe we need to fix the data model instead of doing fixups on the way out.

#27 - 06/19/2019 09:21 PM - Tom Clegg

Lucas Di Pentima wrote:

> Reviewing 14287-controller-structure branch (minus lib/controller/*): LGTM. Just wondering if it's really necessary to include the Specimen APIs, its that needed to support preexisting tests?

Thanks!

I just chose Specimen as a template/example -- it's got the usual CRUD methods, permissions, etc., but no type-specific behavior.

The transition I have in mind is something like

1. Implement a few APIs using this strong-types approach (but still proxying to Rails) on an experimental/configurable basis
2. Reassure ourselves that the new way is working
3. Implement the rest of the APIs the same way
4. Reassure ourselves that the new way is working
5. Make the new code the default
6. Remove the old "pass the whole http request through to Rails" code
7. Port APIs from Rails, and implement new ones, in Go

It's possible we'll end up removing specimens/humans/traits rather than port them to Go, but I don't think it makes a huge difference either way so I went with the "assume things will continue as they are" approach.

#28 - 06/20/2019 02:54 PM - Eric Biagiotti

This approach LGTM, just a few comments.

- federation/conn.go - shouldn't all the API function implementations use conn.chooseBackend? Why do some access local directly?
- rpc/conn.go and sdk/go/auth.go packages both have ContextKeyCredentials exported. In the context docs, it is recommended to not export context keys.
- rpc/conn.go ln 75 & 81. My linter complains: "should not use basic type string as key in context.WithValue". The WithValue doc has a bit more detail on why.

#29 - 06/24/2019 03:23 PM - Tom Clegg

Peter Amstutz wrote:

> router/checker.go
>
> I started going alphabetically and opened this file first and was confused as to what it was for (I guess it is used in testing). Could use a comment at the top.

Yes, it's a gocheck checker. Added comment and renamed to checker_test.go.

> router/error.go
>
> There's no standard struct for errorWithStatus ? This looks very familiar (but maybe I'm just thinking of earlier iterations of controller.)

TransactionError in sdk/go/arvados satisfies the relevant interface{HTTPStatus()int}, but it didn't seem quite right to use that for internal errors too. I figured the important thing was to use the same interface, so it's OK to have multiple error types.

> router/router.go
>
> loadRequestParams() I don't see handling for application/x-www-form-urlencoded but I think that's because req.Form already parses it for us. Could you add a comment?

Yes, that's right. Added comments about req.Form and type-guessing there.

> [...]
>
> There's a number of read-only fields like this, why filter these two in particular? (For example, at one point workbench2 was doing read-modify-update and sending back collection_version which resulted in an error.)

These were the ones that made tests fail. Do you think we need collection_version too here, or is workbench2 fixed?

I don't really like this approach. I'm thinking it would be better to fix the clients (only send the attrs you're trying to change) and/or make the API more lenient at transaction time (allow a no-op update to a read-only field) rather than silently ignore that part of the update.

Etag is special in that the only (?) sensible way to interpret it in an update request is a different spelling of "If-Match", i.e., "abort update if target has already changed underneath me".

For now I'm just hoping to avoid the rabbit hole.

> [...]
>
> Instead of a large, deeply nested anonymous function, it would probably be more maintainable to put the body in its own function.

Done. Indeed, this makes the "accept PUT as synonym for PATCH" bit seem more reasonable.

> Instead of transcode() doing a full serialize/deserialize could we use something like mapstruct instead? (The actual goal seems to be to copy fields between struct and map[string]interface{}). This JSON serialize approach likely creates multiple copies of keys and values (which could be a concern for large fields like mounts or manifest_text).

Yes, there's certainly room for improvement here. I'm not sure it's enough of a bottleneck to merit optimization right now, though. My thinking is to focus on getting something here that works, then port APIs to get the Nginx->Passenger->RailsAPI round-trip out of the picture, then optimize the request/response handling. Does this seem reasonable?

> sendResponse()
>
> Is the ResponseWriter primed to send the "Content-Type: application/json" header? I don't see it (or any other headers) being set.

Ah, looks like not. I suspect this was relying on Go to sniff the response data. Fixed.

> This seems to have three or four strategies to infer what the 'kind' field should be, which feels hacky. Is there a better way to propagate/preserve that information from whatever generated the response originally? If we're having round-tripping problems propagating API responses then maybe we need to fix the data model instead of doing fixups on the way out.

My main goal here was to avoid loading up the model types (arvados.Collection etc) with HTTP-API-v1-response behavior. Like the transcoding stuff, I'm sure there's a better way to do it -- it seems like we should be looking at the types before serializing to JSON instead of working backwards -- but for now I just wanted to get the job done and keep the code relegated to the HTTP responder.

14287-controller-structure @ 03977060ba2024b4f3bbc0146726b609f1915caf --
https://ci.curoverse.com/view/Developer/job/developer-run-tests/1339/

#30 - 06/25/2019 06:01 PM - Peter Amstutz

Tom Clegg wrote:

> > There's a number of read-only fields like this, why filter these two in particular? (For example, at one point workbench2 was doing read-modify-update and sending back collection_version which resulted in an error.)
>
> These were the ones that made tests fail. Do you think we need collection_version too here, or is workbench2 fixed?

I think workbench2 was fixed but only for the special case of collection_version and not in a systematic way that distinguishes the fields the user actually intended to update. I think the main problem is that the API behaves somewhat inconsistently on this point, there are fields that can't be updated, but if you send them in the request it doesn't complain, it just ignores/overrides the value.

> My main goal here was to avoid loading up the model types (arvados.Collection etc) with HTTP-API-v1-response behavior. Like the transcoding stuff, I'm sure there's a better way to do it -- it seems like we should be looking at the types before serializing to JSON instead of working backwards -- but for now I just wanted to get the job done and keep the code relegated to the HTTP responder.

Yea, I suspect when we add the rest of the types to the "API" interface we're going to want to generalize so as to avoid piling up more special cases in sendResponse.

It looks like ARVADOS_EXPERIMENTAL is not set by default. Is it set on Jenkins? How do we intend to validate the new code path?

Rest LGTM.

#31 - 06/25/2019 06:17 PM - Tom Clegg

Eric Biagiotti wrote:

> > - federation/conn.go - shouldn't all the API function implementations use conn.chooseBackend? Why do some access local directly?

CollectionProvenance and CollectionUsedBy: I've updated these to use chooseBackend too. It looks like the existing controller code doesn't support that, but it's so easy we might as well.

List: Oops. It looks like I missed the "federated multi-object query", /arvados/v1/collections?count=none&filters=[["uuid","in",["x","y","z"]]] -- the tests for that only hit /containers/ and the new code only handles /collections/, so I didn't get caught. I think I could merge this branch without it, but it does need to be added before we try EnableBetaController14287 on dev clusters.

- rpc/conn.go and sdk/go/auth.go packages both have ContextKeyCredentials exported. In the context docs, it is recommended to not export context keys.

I think the main rule is that the types should be private, which is enough to prevent collisions (the docs do hint that exported keys are a thing -- "exported context key variables' static type should be a pointer or interface") but the use of NewContext()/FromContext() wrapper functions is explicitly encouraged so I've switched to that instead of exporting the key.

(The rpc/conn.go context key wasn't even used, so I just deleted it.)

- rpc/conn.go ln 75 & 81. My linter complains: "should not use basic type string as key in context.WithValue". The WithValue doc has a bit more detail on why.

Switched this to use a ContextWithAuthorization() func instead.

It turns out using a different empty-struct type per context key is a convenient/idiomatic way to avoid allocations (https://github.com/golang/go/issues/17826#issuecomment-258946985) so I figured I might as well do that here too, instead of using strings.

#32 - 06/25/2019 06:25 PM - Tom Clegg

Peter Amstutz wrote:

> I think workbench2 was fixed but only for the special case of collection_version and not in a systematic way that distinguishes the fields the user actually intended to update. I think the main problem is that the API behaves somewhat inconsistently on this point, there are fields that can't be updated, but if you send them in the request it doesn't complain, it just ignores/overrides the value.

I suppose the issue at hand is whether the controller needs to do anything. It seems that if RailsAPI accepts unchanged values, then controller shouldn't need to remove them. I fixed various bits of Go code that passed an entire Collection{} instead of just the desired set of fields so maybe this isn't even needed any more. I'll check.

> It looks like ARVADOS_EXPERIMENTAL is not set by default. Is it set on Jenkins? How do we intend to validate the new code path?

That's right. The plan is to add ARVADOS_EXPERIMENTAL as a Jenkins build parameter so we can do stuff like "run a dev-tests job on branch B with experimental=foobar" and "run a weekly run-tests job on master with experimental=bazwaz". The rest of the jobs continue to test the recommended-for-production code.

#33 - 06/26/2019 06:33 PM - Peter Amstutz

LGTM.

#34 - 06/26/2019 08:21 PM - Tom Clegg

14287-controller-structure @ 3a28574402bbeb5df3ea8f32f2f60a7a2f20e4fa

#35 - 06/26/2019 08:44 PM - Eric Biagiotti

LGTM also!

#36 - 07/03/2019 03:11 PM - Tom Clegg

- Target version changed from 2019-07-03 Sprint to 2019-07-17 Sprint

#37 - 07/03/2019 03:11 PM - Tom Clegg

- Story points changed from 2.0 to 0.5

#38 - 07/11/2019 03:17 PM - Tom Clegg

14287-federated-list @ b3a016e9a47d453b5ae4d287d8b6eaafd69971df -- https://ci.curoverse.com/view/Developer/job/developer-run-tests/1385/

...implements the "federated list" behavior I missed (see note-28, note-31)

#39 - 07/11/2019 07:31 PM - Eric Biagiotti

Tom Clegg wrote:

> 14287-federated-list @ b3a016e9a47d453b5ae4d287d8b6eaafd69971df -- https://ci.curoverse.com/view/Developer/job/developer-run-tests/1385/
>
> ...implements the "federated list" behavior I missed (see note-28, note-31)

Workbench tests failed and not with the typical PhantomJS crash.

source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L133 Commented out code
source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L143 Always helpful to have a comment upfront describing nested maps
source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L158 Should be 'uuid = ...' **OR** 'uuid in [...]
source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L184 If this isn't redundant, I'm assuming it has something to do with using these variables in the go func. Would different variable names be more clear or is this a common pattern in Go?

#40 - 07/17/2019 02:48 PM - Tom Morris

- Target version changed from 2019-07-17 Sprint to 2019-07-31 Sprint

#41 - 07/22/2019 05:18 PM - Tom Clegg

> Workbench tests failed and not with the typical PhantomJS crash.

Lots of timeouts. I suspect this is just a newer kind of wb flakiness. Trying again.

> source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L133 Commented out code

That was pseudocode, intended to explain the following block. I've reworded it so it (hopefully) looks more like a comment and less like commented-out code.

> source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L143 Always helpful to have a comment upfront describing nested maps

Added.

> source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L158 Should be 'uuid = ...' **OR** 'uuid in [...]

Wouldn't that imply you can only do "=" or "in", but not both? Maybe this needs different phrasing, like "each filter in a federated list query must be either "uuid = ..." or "uuid in [...]""?

> source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L184 If this isn't redundant, I'm assuming it has something to do with using these variables in the go func. Would different variable names be more clear or is this a common pattern in Go?

It makes a new variable for each goroutine (otherwise all the goroutines incorrectly share the same loop variable). I've changed this to pass the values explicitly as arguments to the inline func.

14287-federated-list @ 83ee4d3ebf739f8aad67cf8faea093fc71d24d01 -- https://ci.curoverse.com/view/Developer/job/developer-run-tests/1407/

#42 - 07/23/2019 03:12 PM - Eric Biagiotti

Tom Clegg wrote:

> > source:/lib/controller/federation/list.go@b3a016e9a47d453b5ae4d287d8b6eaafd69971df#L158 Should be 'uuid = ...' **OR** 'uuid in [...]
>
> Wouldn't that imply you can only do "=" or "in", but not both? Maybe this needs different phrasing, like "each filter in a federated list query must be either "uuid = ..." or "uuid in [...]""?

"Either/or" is mutually exclusive, "or" can mean both, but it isn't clear. How about "each filter in a federated list query must be either "uuid = ...", "uuid in [...]", or both"?

This also needs to be updated in the comment on line 71.

Other than that, this LGTM.

#43 - 07/23/2019 03:30 PM - Tom Clegg

Eric Biagiotti wrote:

> How about "each filter in a federated list query must be either "uuid = ...", "uuid in [...]", or both"?

Done. (But deleted "or both" because a filter can't be both at once.)

> This also needs to be updated in the comment on line 71.

Done.

14287-federated-list @ f0451bbb69fd79fbead3036aac29dd97977727a6 -- https://ci.curoverse.com/view/Developer/job/developer-run-tests/1413/

#44 - 07/31/2019 03:00 PM - Tom Clegg

- Target version changed from 2019-07-31 Sprint to 2019-08-14 Sprint

#45 - 07/31/2019 03:07 PM - Tom Clegg

- Status changed from In Progress to Resolved

- Target version changed from 2019-08-14 Sprint to 2019-07-31 Sprint

#46 - 10/31/2019 03:39 PM - Tom Clegg

- Status changed from Resolved to In Progress

- Target version changed from 2019-07-31 Sprint to 2019-11-06 Sprint

Enabled on 4xphq in order to test #15107 and found bug in list handler.

14287-federated-list @ ee92b80ba453d7669614e258d2f4ea639516a77f -- https://ci.curoverse.com/view/Developer/job/developer-run-tests/1621/

#47 - 10/31/2019 06:59 PM - Peter Amstutz

Tom Clegg wrote:

> Enabled on 4xphq in order to test #15107 and found bug in list handler.
>
> 14287-federated-list @ ee92b80ba453d7669614e258d2f4ea639516a77f --
> https://ci.curoverse.com/view/Developer/job/developer-run-tests/1621/

Forgetting the base case! Classic mistake. This LGTM.

#48 - 10/31/2019 07:04 PM - Tom Clegg

- Status changed from In Progress to Resolved

Applied in changeset arvados|d7c47dc44a935fe99469e1c7549ad8154ed0408f.

#49 - 12/04/2019 07:43 PM - Tom Clegg

- Related to Story #15922: Change EnableBetaController14287 (default false) to ForceLegacyAPI14 (default false) added