Arvados - Story #3021

[Tests] Make jenkins tests run faster.

06/13/2014 02:46 PM - Tom Clegg

Status:	Resolved	**Start date:**	01/05/2015
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	100%
Category:	Tests	**Estimated time:**	0.00 hour
Target version:	2015-02-18 sprint		

Description

Places to look include:

- Performance of Workbench itself. Are some tests simply revealing performance problems in the application?
- API server setup/teardown during workbench tests. (This only happens once per suite since rails4, so perhaps it's not a big contributor.)
- API server setup/teardown during python tests.

Tools to fix this include:

- API server hook that resets test fixtures to a pristine state. Python/Workbench integration tests should use this instead of starting and stopping API server many times.
- Refactor "run_test_server" to run as a wrapper around an arbitrary test suite. It can set an environment variable to communicate to the test suite how to push the "reset fixtures" button (path to a trigger file, PID to send a USR1 signal to, or whatever).

Subtasks:

Task # 4971: Review 3021-test-perf on arvados-dev	**Resolved**
Task # 4896: Review 3021-more-phantomjs	**Resolved**
Task # 4895: Convert selenium tests to phantomjs where possible	**Resolved**
Task # 4902: Use a fixture instead of calling "run a pipeline" in the set of 8 "rerun p...	**Resolved**
Task # 4974: Review 3021-workbench-perf	**Resolved**
Task # 4975: Review 3021-api-performance	**Resolved**
Task # 5092: Review branch: 4902-use-fixture-to-rerun-pipelines	**Resolved**
Task # 4986: Refactor run_test_server.py to use API database reset rather than restarti...	**Resolved**
Task # 5130: Review 3021-leave-api-running (arvados-dev)	**Resolved**
Task # 5129: Review 3021-leave-api-running (arvados)	**Resolved**

Related issues:

Related to Arvados - Bug #4942: [Workbench] Data Collections Tab in Projects ...	**Closed**	**01/08/2015**
Related to Arvados - Bug #4464: [Workbench] Collections tab loads forever on ...	**Resolved**	**02/05/2015**
Related to Arvados - Feature #2660: [Tests] [Draft] Make entire test suite ru...	**Resolved**	
Related to Arvados - Feature #4534: [Tests] run-tests.sh should bring up API ...	**Resolved**	**11/20/2014**

Associated revisions

Revision b5213861 - 01/07/2015 05:14 PM - Tom Clegg

Merge branch '3021-more-phantomjs' refs #3021

Revision 3636b3a3 - 01/08/2015 09:22 PM - Tom Clegg

3021: Fix phantomjs races by waiting for pages to appear. refs #3021

Revision 2fad64f0 - 01/09/2015 10:45 PM - Tom Clegg

Diagnostics really do need selenium. refs #3021

Revision 61fdce2e - 01/09/2015 10:46 PM - Tom Clegg

Add a magic pseudoclass to body, instead of appending a magic div. Selenium seems to like this better. refs #3021

Revision 20354a24 - 01/13/2015 10:10 PM - Tom Clegg

Merge branch '3021-test-perf' refs #3021

Revision 20354a24 - 01/13/2015 10:10 PM - Tom Clegg

Merge branch '3021-test-perf' refs #3021

Revision 3fec99e7 - 01/16/2015 06:37 AM - Tom Clegg

Merge branch '3021-workbench-perf' refs #3021

Revision 8d61b342 - 01/16/2015 05:26 PM - Tom Clegg

Update bundle. refs #3021

Revision 224701a0 - 01/21/2015 08:03 PM - Tom Clegg

Merge branch '3021-api-performance' refs #3021

Revision 6772d21b - 02/06/2015 04:55 PM - Brett Smith

3021: Remove obsolete accommodation for buggy python-daemon.

This bug has been fixed in newer versions. Refs #3021.

Revision 6772d21b - 02/06/2015 04:55 PM - Brett Smith

3021: Remove obsolete accommodation for buggy python-daemon.

This bug has been fixed in newer versions. Refs #3021.

Revision afef0760 - 02/06/2015 10:30 PM - Tom Clegg

Merge branch '3021-leave-api-running' refs #3021

Revision d7f1a63c - 02/06/2015 10:58 PM - Tom Clegg

Merge branch '3021-leave-api-running' refs #3021

Revision d7f1a63c - 02/06/2015 10:58 PM - Tom Clegg

Merge branch '3021-leave-api-running' refs #3021

History

#1 - 06/17/2014 03:10 PM - Tom Clegg

- Description updated

#2 - 06/18/2014 03:34 PM - Tom Clegg

- Target version deleted (2014-07-16 Sprint)

#3 - 07/04/2014 04:03 PM - Tom Clegg

- Target version set to Arvados Future Sprints

#4 - 07/04/2014 04:04 PM - Tom Clegg

- Subject changed from Make jenkins tests run faster. to [Tests] Make jenkins tests run faster.

#5 - 01/04/2015 08:56 AM - Tom Clegg

- Category set to Tests

- Status changed from New to In Progress

3021-more-phantomjs @ d43df86

vcpu	GHz	bogomips	VM	disk	workbench tests	clock[1]
4x AMD Phenom(tm) II X6 1090T	3.2	6429	Xen 4.1.4-3+deb7u3	HDD RAID-1	Finished in 1025.875247s Finished in 1273.093580s @master[2]	17m37.723s 21m43.878s @master[2]

2x Intel(R) Core(TM) i5-2430M	2.4	4789	VirtualBox 4.3.10-dfsg-1	SSD	Finished in 1538.421145s Finished in 2005.847655s @master[2]	26m0.652s 33m45.994s @master[2]

[1] time run-tests [...] --skip-install --only apps/workbench

[2] master @ c4e7c6d

#6 - 01/05/2015 03:53 PM - Tom Clegg

- Target version changed from Arvados Future Sprints to 2015-01-07 sprint

#7 - 01/05/2015 06:57 PM - Brett Smith

Reviewing d43df86. The branch looks great overall, and I'm looking forward to seeing this merged. Thanks for tackling this.

Unfortunately, I'm getting seemingly random errors in the pipeline instances tests. They all look like this:

```
  1) Failure:
PipelineInstancesTest#test_Rerun_pipeline_instance_as_active_using_options_true_false_in_false [/home/brett/repos/arvados/apps/workbench/test/integration/pipeline_instances_test.rb:319]:
Rerun instance path expected to be different.
Expected "/pipeline_instances/zzzzz-d1hrv-0w8ekwuvml73qg8" to not be equal to "/pipeline_instances/zzzzz-d1hrv-0w8ekwuvml73qg8".
```

First I ran all the tests, and got two failures, for active/true_false_in_false and active/true_true_in_true. Then I ran just this test file, and got those plus a new error, for active/true_true_in_false. The "URL not equal" assertion is always the one that fails.

On Headless invocation: if I'm following right, the current theory is:

1. On an idle host, a test suite starts. It creates a new X session for itself.
2. While that's running, a second test suite starts. It finds the existing X session and reuses it.
3. The first test suite finishes, and shuts down the X session. Tests in the second suite fail from here on out, because their X server disappeared.

If that's the right theory, I think this is promising for solving our ECONNREFUSED issue, but I'm not sure it goes as far as necessary. Looking at the source documentation, it seems to me like what we want is just reuse: false, without specifying a display number—that way each test session will start a new X server that lives exactly as long as it does. How does that sound to you?

Whatever rule we decide, I think we need to make sure our Headless starts in the other test suites (diagnostics+performance) follow it, to make sure no two suites can step on each others' toes.

Thanks.

#8 - 01/06/2015 03:00 PM - Tom Clegg

Brett Smith wrote:

> First I ran all the tests, and got two failures, for active/true_false_in_false and active/true_true_in_true. Then I ran just this test file, and got those plus a new error, for active/true_true_in_false. The "URL not equal" assertion is always the one that fails.

Hm, I admit I don't quite see *how* the headless/driver stuff causes this failure. (As an aside, it looks like this set of 8 "rerun pipeline instance" tests is somewhat wasteful: each one seems to call the expensive "run a pipeline" procedure merely to create [what should be] fixtures. I'll add that to the list of performance todos.)

> Looking at the source documentation, it seems to me like what we want is just reuse: false, without specifying a display number—that way each test session will start a new X server that lives exactly as long as it does. How does that sound to you?

I think you're right. I've rearranged the helper stuff (as a module, with a Headless singleton) and used

- create one Headless object per "rake test" process
- reuse: false
- headless.start at the beginning of a test (if the test is using it)
- headless.stop at the end of a test (if we started it)

> Whatever rule we decide, I think we need to make sure our Headless starts in the other test suites (diagnostics+performance) follow it, to make sure no two suites can step on each others' toes.

Ah, indeed. Updated these too. (They don't seem to need selenium, either.)

Now at 7b5c84e

#9 - 01/06/2015 05:08 PM - Brett Smith

- File workbench-fail-1.png added

Tom Clegg wrote:

> Brett Smith wrote:
>
> > First I ran all the tests, and got two failures, for active/true_false_in_false and active/true_true_in_true. Then I ran just this test file, and got those plus a new error, for active/true_true_in_false. The "URL not equal" assertion is always the one that fails.
>
> Hm, I admit I don't quite see *how* the headless/driver stuff causes this failure. (As an aside, it looks like this set of 8 "rerun pipeline instance" tests is somewhat wasteful: each one seems to call the expensive "run a pipeline" procedure merely to create [what should be] fixtures. I'll add that to the list of performance todos.)

Rerunning for the current branch, I got basically the same errors, but with another new set: active/true_false_in_false, active/true_true_in_false, and active/true_true_in_true. I took a look at the screenshots (attached), and they show that the test is still on the re-run options modal. It hasn't moved on to the new pipeline page.

A few things catch my attention:

- It's pretty likely a race condition, since the set of failing tests keeps shifting.
- These tests changed drivers from Capybara to Poltergeist.
- The tests that fail consistently have with_options set, meaning that it's going through the re-run options modal submission form.
- The test clicks the modal button to create the new pipeline, then immediately asserts the URL, with nothing in between.

Is there any chance that the Poltergeist driver is slower to update current_path on form submissions than Capybara? That would explain all the issues we've seen, I think.

#10 - 01/06/2015 05:56 PM - Tom Clegg

Brett Smith wrote:

> - The test clicks the modal button to create the new pipeline, then immediately asserts the URL, with nothing in between.
>
> Is there any chance that the Poltergeist driver is slower to update current_path on form submissions than Capybara? That would explain all the issues we've seen, I think.

This seems quite plausible. I still haven't seen any of these failures at my end, so I can only guess whether it helps, but I've added a "wait for dialog to close" statement in 9c10212.

#11 - 01/06/2015 10:10 PM - Brett Smith

Tom Clegg wrote:

> This seems quite plausible. I still haven't seen any of these failures at my end, so I can only guess whether it helps, but I've added a "wait for dialog to close" statement in 9c10212.

Unfortunately, this just makes the tests fail on that assertion. The problem is on the other end: we're working on the modal dialog when it's transitioning in. I think what happens is that .click calculates a position for the element, moves the mouse cursor there, then generates the click event. However, the ongoing transition means that the element is no longer under the mouse when the click event is generated.

Adding sleep(0.5) before within('.modal-dialog') resolves the issue for me. Reviewing the Bootstrap documentation, I don't see a better way to be sure that the transition is done before we start interacting with the modal. Several classes and other attributes are changed, but they seem to be set when the transition starts. I tried asserting body.modal-open instead of sleeping; that was not sufficient. If you can find a better alternative, great; but one way or another, I think we should replace the post-assertion with waiting for the transition.

Then, a little belated, but a couple of comments about the Headless cleanup in integration_helper.rb:

- The file's toplevel still has a line Headless.new.start, which I believe is redundant with the addition of HeadlessHelper.
- Should this file require 'headless'? I'm not sure about the exact boundaries of Rails' library loading, so I could well be wrong.

Thanks.

#12 - 01/06/2015 10:58 PM - Tom Clegg

Brett Smith wrote:

> If you can find a better alternative, great; but one way or another, I think we should replace the post-assertion with waiting for the transition.

OK, I've added (in 76ab57b) a helper that waits for a shown.bs.modal event after clicking the "Re-run options" button. Bootstrap docs say:

"This event is fired when the modal has been made visible to the user (will wait for CSS transitions to complete)."

And bootstrap.js agrees/clarifies "will wait for...to complete" means "the event does not fire until..."

- The file's toplevel still has a line Headless.new.start, which I believe is redundant with the addition of HeadlessHelper.

Oops, yes, removed.

- Should this file require 'headless'? I'm not sure about the exact boundaries of Rails' library loading, so I could well be wrong.

I'm not 100% clear about this part myself but I think its presence in Gemfile (without require:false) causes the Bundler.require() in application.rb to load it for us.

#13 - 01/07/2015 02:06 PM - Brett Smith

Tom Clegg wrote:

> OK, I've added (in 76ab57b) a helper that waits for a shown.bs.modal event after clicking the "Re-run options" button.

While I was writing my last comment, I had this same train of thought: "Waiting for the shown.bs.modal event to fire would work. But I don't see a way to listen for that with Capybara without modifying the DOM directly." I stopped the train there, because I feel a little uneasy about modifying the DOM while it's under testing. It feels a little like messing with a class' private instance variables while you're testing it: your tests get further away from testing the system's real behavior, because your private machinations might put the system in a state that it can't or won't actually get to in normal use.

But maybe this is what we have to do to work around limitations of Bootstrap/Capybara. Definitely I think you've implemented it in the nicest possible way. (My one bikesheddy comment is that it might be nice to generate magic randomly every time, but that's trivial.) I'm comfortable with merging 76ab57bb, if you're comfortable with the approach. Thanks.

#14 - 01/07/2015 04:42 PM - Tom Clegg

Brett Smith wrote:

> I feel a little uneasy about modifying the DOM while it's under testing. It feels a little like messing with a class' private instance variables while you're testing it: your tests get further away from testing the system's real behavior, because your private machinations might put the system in a state that it can't or won't actually get to in normal use.

This would be a bit better if it used something more isolated than the DOM, like window.privateTestingState.gotThatEvent=true + https://github.com/begriffs/capybara-js... Meanwhile we're counting on the app not to care that an event handler and a div get added and removed.

> But maybe this is what we have to do to work around limitations of Bootstrap/Capybara.

Yes, I think of this as an implementation-dependent proxy for "wait until things stop moving around on the screen". It would be better if the testing framework took care of this for us.

> (My one bikesheddy comment is that it might be nice to generate magic randomly every time, but that's trivial.)

I initially used random, but then thought static-and-ugly would be easier to debug. "{staticpart}-{randompart}" OK? dea2858

#15 - 01/07/2015 04:57 PM - Brett Smith

Tom Clegg wrote:

> Brett Smith wrote:
>
> > (My one bikesheddy comment is that it might be nice to generate magic randomly every time, but that's trivial.)
>
> I initially used random, but then thought static-and-ugly would be easier to debug. "{staticpart}-{randompart}" OK? dea2858

Sure, that's fine too. Go ahead and merge dea2858. Thanks.

#16 - 01/07/2015 08:08 PM - Tom Clegg

- Target version changed from 2015-01-07 sprint to Arvados Future Sprints

#17 - 01/13/2015 02:53 PM - Tom Clegg

- Target version changed from Arvados Future Sprints to 2015-01-28 Sprint

#18 - 01/13/2015 04:36 PM - Brett Smith

Reviewing arvados-dev commit:0a471810

Everything looks good, I just have one sort of tangential comment. The point of easy_install *.egg in do_test is to automatically install a package's test dependencies (e.g., PyYAML) for others to use. Since we've pretty officially given up on that approach and install them all up front, removing this line might help avoid surprises down the line. Especially since a developer could have any old eggs in their workspace for whatever reason.

But with or without that change, this is good to merge. Thanks.

#19 - 01/14/2015 12:26 AM - Tom Clegg

3021-workbench-perf includes:

- Optimize manifest parsing in Ruby SDK (convert some big arrays to generators).
- Skip "drop/create API database" in Workbench tests. This accounted for most of the ~30s clock time needed to run something like run-tests.sh --only apps/workbench --skip-install apps/workbench_test="TEST=test/unit/ultrafasttest.rb". See related change in arvados-dev|5fb9648
- Refactor user notifications into a helper method, so we don't bother doing the API calls for them unless/until we decide to display them.
- Fix some races in performance tests.
- Add a web-inspectable profiling mode, if ENABLE_PROFILING env var is set. See notes at 288d22d8
- Update dependencies.
  - Relax some needlessly restrictive version pins (sass-rails, simplecov, httpclient).
  - Move minitest into test/diagnostics group.
  - Update bundle. Rails 4.2 needs some changes, so we're still at 4.1.9.

#20 - 01/14/2015 06:29 AM - Tom Clegg

3021-api-performance includes a few performance improvements whose commit messages are much bigger than their code changes:

- 1974e0e Use Oj to encode/decode API responses.
- eec57e3 Don't call Rails.application.eager_load! so ... eagerly.
- f4cd3a9 Don't compute md5(old_attributes.inspect) except when actually doing something with the resulting hash.

They were uncovered by doing some profiling with:

- f2834f4 A tiny performance test class, finally replacing the "test homepage" sample test provided by rails new back in the day.

And an unrelated itty bitty bugfix:

- 2d09568 Fix crash (when testing CORS headers) on missing return_to param. (Not really a performance improvement. Whether you hit this depends on your auth config.)

#21 - 01/14/2015 03:59 PM - Brett Smith

Reviewing 3021-workbench-perf at cd4f506.

Tom Clegg wrote:

> 3021-workbench-perf includes:
>
> - Optimize manifest parsing in Ruby SDK (convert some big arrays to generators).

The first line of each_file_spec refers to now-gone speclist argument.

> - Skip "drop/create API database" in Workbench tests. This accounted for most of the ~30s clock time needed to run something like run-tests.sh --only apps/workbench --skip-install apps/workbench_test="TEST=test/unit/ultrafasttest.rb". See related change in arvados-dev|5fb9648

I know you explained this to me on IRC, so I know I have myself to blame for not following the thread carefully enough, but I'm really not wild about how this means you may be on the hook to do manual setup every time you want to run bundle exec rake test. The downsides of relying on run-tests are:

- It's a separate piece of code in a completely different repository, meaning there's more setup hurdles for a new developer to clear before they can get started.
- run-tests will always incur more overhead than running the tests directly because of how it resets so much of the environment every time. I like that it does that, and it's important, but there are a lot of branches where it's not immediately relevant.
- The run-tests incantation is a lot more verbose than the equivalent rake line.

I would prefer to see individual test runners to take on more responsibility for setting themselves up, rather than less. run-tests will always exist as the documented way to make sure cross-component testing happens correctly, and that's good, but I think in a lot of ways it's a lot more

developer-friendly to make sure it's easy to run individual component tests when you're only working within that component. In this specific case, I would prefer to see necessary database setup move out of run-tests.sh completely, and into Workbench's test scaffolding. If I'm following right, the redundancy is what hurt overall clock time, so making sure it's done exactly once is the key change. Is there some reason that wouldn't achieve the goal?

- Refactor user notifications into a helper method, so we don't bother doing the API calls for them unless/until we decide to display them.

I'm consistently getting this test failure:

```
  1) Error:
ProjectsTest#test_error_while_loading_tab:
ActionView::Template::Error: Errno::ENETUNREACH error connecting to API server
    app/models/arvados_api_client.rb:133:in `rescue in block in api'
    app/models/arvados_api_client.rb:130:in `block in api'
    app/models/arvados_api_client.rb:129:in `synchronize'
    app/models/arvados_api_client.rb:129:in `api'
    app/models/arvados_resource_list.rb:192:in `each_page'
    app/models/arvados_resource_list.rb:113:in `each'
    app/controllers/application_controller.rb:640:in `block in <class:ApplicationController>'
    app/controllers/application_controller.rb:670:in `call'
    app/controllers/application_controller.rb:670:in `block in user_notifications'
    app/controllers/application_controller.rb:669:in `map'
    app/controllers/application_controller.rb:669:in `user_notifications'
    app/views/layouts/body.html.erb:44:in `_app_views_layouts_body_html_erb___3773684380103722667_15027480'
    app/views/layouts/application.html.erb:49:in `_app_views_layouts_application_html_erb__1883942643258282554
_38257040'
    app/controllers/application_controller.rb:55:in `block (2 levels) in render_error'
    app/controllers/application_controller.rb:49:in `render_error'
    app/controllers/application_controller.rb:96:in `render_exception'
```

I believe this is happening because the view for the error page is now trying to contact the API server in a place where it wasn't before. In the old flow, the notification filter would try to fetch notifications, raise an exception, we'd go to render_exception, and that view would work with the (nil) notifications instance variable. Now the original request generates an exception later, goes to render_exception, and the view calls user_notifications, which tries to make a fresh API request that raises its own exception, and then we're in 500 land. If I've got that right, it needs to be addressed somehow.

Thanks.

#22 - 01/15/2015 06:24 PM - Tom Clegg

Brett Smith wrote:

> The first line of each_file_spec refers to now-gone speclist argument.

Oops. Fixed.

> I know you explained this to me on IRC, so I know I have myself to blame for not following the thread carefully enough, but I'm really not wild about how this means you may be on the hook to do manual setup every time you want to run bundle exec rake test.

bundle exec rake test should work fine unless you've dropped/erased your API server's test database for some reason, or you haven't even set it up yet. I think of "database has been created, test fixtures are present" as the normal resting state of a RAILS_ENV=test apiserver. Is this not true for your dev cycle?

> If I'm following right, the redundancy is what hurt overall clock time, so making sure it's done exactly once is the key change. Is there some reason that wouldn't achieve the goal?

Here I was optimizing the "re-run a workbench test until it works" development case, which doesn't require dropping and recreating the API server's database, which is very slow (or even resetting to fixtures, which is less slow).

who	before 3021	after 3021
run-tests.sh apiserver_install	drop db; create db	drop db; create db; load fixtures
workbench test helper	re-create db; load fixtures	noop

> I'm consistently getting this test failure:
>
> I believe this is happening because the view for the error page is now trying to contact the API server in a place where it wasn't before.

Hm, not sure why I didn't catch that (even after wondering about that specific issue while rearranging the code!). Addressed in eb7dde7 by skipping user notifications on error pages.

At first I considered catching/ignoring notification exceptions when showing an error page, but figured it's probably best for everyone if we keep the error-displaying path as simple/short as possible. Also, "hey, try this feature next" seems a little out of place on an error page anyway, if the choice of "this feature" isn't related to fixing the error you just got. Thoughts?

Further discussion of test infrastructure

> The downsides of relying on run-tests are:
>
> - It's a separate piece of code in a completely different repository, meaning there's more setup hurdles for a new developer to clear before they can get started.

We should fix this by putting the test runner in the arvados tree, right?

> - The run-tests incantation is a lot more verbose than the equivalent rake line.

The general solution I've been thinking of is that Workbench (and everyone else) should call a "give me service X" script, which lives in the arvados tree. run-tests should call that script to bring up an API server and leave it running for fuse, workbench, etc. Then, when Workbench says "give me service {API}", the script should recognize that run-tests has already brought one up, and do nothing.

> I would prefer to see individual test runners to take on more responsibility for setting themselves up, rather than less. run-tests will always exist as the documented way to make sure cross-component testing happens correctly, and that's good, but I think in a lot of ways it's a lot more developer-friendly to make sure it's easy to run individual component tests when you're only working within that component.

I agree Workbench should have the responsibility to state "I need an API server", although I'd also say it should have much less *code*. For example, it seems awful to me that Workbench's test suite needs to know which http server is most suitable for running an API with/without websockets, and how to generate self-signed certs for apiserver.

> In this specific case, I would prefer to see necessary database setup move out of run-tests.sh completely, and into Workbench's test scaffolding.

One thing I think we should be getting from run-tests.sh is an assurance that there exists (at least) one way of installing API server such that all of the client software works against that single instance. I believe the above strategy, where everyone uses the same script to bring up servers, lets us have it both ways -- and lets us make optimizations/features like "leave same API server running while testing multiple clients" and "leave servers running after tests, so I can debug" and "test database already created, no need to re-create now" without having N copies of that logic.

Then we won't care so much whether a given action like db:setup is an "install" or a "startup" task. Workbench will just demand an API server with test fixtures, and won't need to know which steps are needed to make that happen.

#23 - 01/15/2015 08:24 PM - Brett Smith

At eb7dde7, I am consistently getting these test failures. It looks like something is interfering with our textarea editable rendering. Looking at the screenshots, these are popping up as plain input fields.

```
  1) Error:
PipelineInstancesTest#test_pipeline_description:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
    test/integration/pipeline_instances_test.rb:164:in `block (2 levels) in <class:PipelineInstancesTest>'
    test/integration/pipeline_instances_test.rb:162:in `block in <class:PipelineInstancesTest>'
    test/test_helper.rb:254:in `run'

  2) Error:
ProjectsTest#test_Find_a_project_and_edit_description_to_html_description:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
    test/integration/projects_test.rb:71:in `block (3 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:69:in `block (2 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:67:in `block in <class:ProjectsTest>'
    test/test_helper.rb:254:in `run'

  3) Error:
ProjectsTest#test_Find_a_project_and_edit_description_to_textile_description:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
    test/integration/projects_test.rb:42:in `block (3 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:40:in `block (2 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:38:in `block in <class:ProjectsTest>'
    test/test_helper.rb:254:in `run'

  4) Error:
ProjectsTest#test_Find_a_project_and_edit_description_to_textile_description_with_link_to_object:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
```

```
    test/integration/projects_test.rb:94:in `block (3 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:92:in `block (2 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:90:in `block in <class:ProjectsTest>'
    test/test_helper.rb:254:in `run'

  5) Error:
ProjectsTest#test_Find_a_project_and_edit_its_description:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
    test/integration/projects_test.rb:24:in `block (3 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:22:in `block (2 levels) in <class:ProjectsTest>'
    test/integration/projects_test.rb:20:in `block in <class:ProjectsTest>'
    test/test_helper.rb:254:in `run'

  6) Error:
PipelineTemplatesTest#test_pipeline_template_description:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
    test/integration/pipeline_templates_test.rb:28:in `block (2 levels) in <class:PipelineTemplatesTest>'
    test/integration/pipeline_templates_test.rb:26:in `block in <class:PipelineTemplatesTest>'
    test/test_helper.rb:254:in `run'

  7) Error:
JobsTest#test_add_job_description:
Capybara::ElementNotFound: Unable to find css ".editable-input textarea"
    test/integration/jobs_test.rb:28:in `block (2 levels) in <class:JobsTest>'
    test/integration/jobs_test.rb:26:in `block in <class:JobsTest>'
    test/test_helper.rb:254:in `run'
```

Tom Clegg wrote:

> bundle exec rake test should work fine unless you've dropped/erased your API server's test database for some reason, or you haven't even set it up yet. I think of "database has been created, test fixtures are present" as the normal resting state of a RAILS_ENV=test apiserver. Is this not true for your dev cycle?

I'm often in that state, but then life messes it up:

- I pull new API server migrations and haven't run them yet.
- I pull new API server test fixtures and haven't loaded them yet.
- I run another test suite that modifies the test database and leaves it dirty. (I know making those reset-aware is on the todo list, but still.)

> Here I was optimizing the "re-run a workbench test until it works" development case, which doesn't require dropping and recreating the API server's database, which is very slow (or even resetting to fixtures, which is less slow).

I guess one way to express my worry here is: will this provide net time savings? Or will we just trade time, speeding up repeat runs at the cost of more frequently babysitting the test database in situations like above?

(This is not my biggest worry. If most people have workflows that better avoid this problem, I'm happy to accept that I'm the outlier and adjust accordingly.)

> At first I considered catching/ignoring notification exceptions when showing an error page, but figured it's probably best for everyone if we keep the error-displaying path as simple/short as possible. Also, "hey, try this feature next" seems a little out of place on an error page anyway, if the choice of "this feature" isn't related to fixing the error you just got. Thoughts?

I find the second argument pretty compelling. If you try to add an SSH key to your account and land on an error page, a notification telling you, "Hey, you should upload an SSH key!" adds insult to injury.

Further discussion of test infrastructure

> The downsides of relying on run-tests are:
>
> - It's a separate piece of code in a completely different repository, meaning there's more setup hurdles for a new developer to clear before they can get started.
>
> We should fix this by putting the test runner in the arvados tree, right?

That would be an improvement, but it's still not ideal. "To run the tests, you have to use this separate test script elsewhere in the repository" is still a learning hurdle, compared to "To run the tests, run bundle exec rake, just like every other well-behaved Bundle+Rails Web application."

- The run-tests incantation is a lot more verbose than the equivalent rake line.

> The general solution I've been thinking of is that Workbench (and everyone else) should call a "give me service X" script, which lives in the arvados tree. run-tests should call that script to bring up an API server and leave it running for fuse, workbench, etc. Then, when Workbench says "give me service {API}", the script should recognize that run-tests has already brought one up, and do nothing.

This is the "generalized and improved run_test_server.py" idea we've talked about before, right? I like the sound of that too. And I like it because our individual test suites can talk to it directly, enabling the user to run tests in the normal way, rather than requiring the user to know what setup they need to do before running a test suite. I attached a task to capture this idea.

Thanks.

#24 - 01/15/2015 09:54 PM - Tom Clegg

Brett Smith wrote:

> At eb7dde7, I am consistently getting these test failures. It looks like something is interfering with our textarea editable rendering. Looking at the screenshots, these are popping up as plain input fields.

What the. Trying to reproduce here.

- I pull new API server migrations and haven't run them yet.
- I pull new API server test fixtures and haven't loaded them yet.

These things need to happen around the same time "bundle install" needs to happen. Should Workbench (via run_test_servers.py or otherwise) run apiserver's "bundle install", too? (I've run into this one before, but I figured it was my own fault for not updating, not Workbench's test suite's fault for not knowing how to update.)

- I run another test suite that modifies the test database and leaves it dirty. (I know making those reset-aware is on the todo list, but still.)

Hm, a "reset to fixtures" call would address this (without being slow). We already do it before every subsequent test case, after all, so it seems like we should do it before the first, too.

> > Here I was optimizing the "re-run a workbench test until it works" development case, which doesn't require dropping and recreating the API server's database, which is very slow (or even resetting to fixtures, which is less slow).
>
> I guess one way to express my worry here is: will this provide net time savings? Or will we just trade time, speeding up repeat runs at the cost of more frequently babysitting the test database in situations like above?

My approach has been to use either "fastest possible run, for targeted test when I think I only changed one little thing" or "shortcut not available because migrate/bundle/whatever, so don't --skip-install" mode.

It sounds like we both want the same thing, except

- Running bundle exec rake test from apps/workbench should -- one way or another -- do all the same stuff run-tests does now, so tests actually work (we both want this, the difference is just that it already doesn't work for me, so I always use run-tests.sh)
- You aren't inclined to use a "skip install" option, because of the babysitting risk.

How about this:

```
unless ENV['ARVADOS_TEST_API_INSTALLED']
  _system('bundle', 'exec', 'rake', 'db:test:load')
  _system('bundle', 'exec', 'rake', 'db:fixtures:load')
end
```

With this sort of pattern, run-tests.sh can set ARVADOS_TEST_API_INSTALLED (to a path), and Workbench will save some time. Next ARVADOS_TEST_API_HOST can point to its ip:port, and Workbench/run_test_server.py can skip their "start up an API server" stuff.

And eventually everything will pay attention to the ip:port and we'll be able to run two test suites on a single host without port number catastrophe.

> > At first I considered catching/ignoring notification exceptions when showing an error page, but figured it's probably best for everyone if we keep the error-displaying path as simple/short as possible. Also, "hey, try this feature next" seems a little out of place on an error page anyway, if the choice of "this feature" isn't related to fixing the error you just got. Thoughts?
>
> I find the second argument pretty compelling. If you try to add an SSH key to your account and land on an error page, a notification telling you, "Hey, you should upload an SSH key!" adds insult to injury.

Exactly. Or, "Hm, your token expired" + "You should upload an SSH key!" → uh, will that really give me a fresh token? ok, I guess I'll try it...

> We should fix this by putting the test runner in the arvados tree, right?

That would be an improvement, but it's still not ideal. "To run the tests, you have to use this separate test script elsewhere in the repository" is still a learning hurdle, compared to "To run the tests, run bundle exec rake, just like every other well-behaved Bundle+Rails Web application."

Right. I didn't mean to exclude the option of running "rake test". I just meant it would make sense if you also had an option like "git clone arvados.git; cd arvados; make test" (that would test everything).

> The general solution I've been thinking of is that Workbench (and everyone else) should call a "give me service X" script, which lives in the arvados tree. run-tests should call that script to bring up an API server and leave it running for fuse, workbench, etc. Then, when Workbench says "give me service {API}", the script should recognize that run-tests has already brought one up, and do nothing.

This is the "generalized and improved run_test_server.py" idea we've talked about before, right? I like the sound of that too. And I like it because our individual test suites can talk to it directly, enabling the user to run tests in the normal way, rather than requiring the user to know what setup they need to do before running a test suite. I attached a task to capture this idea.

Yes, exactly. Thanks.

#25 - 01/15/2015 10:42 PM - Brett Smith

Tom Clegg wrote:

> Brett Smith wrote:
>
> > At eb7dde7, I am consistently getting these test failures. It looks like something is interfering with our textarea editable rendering. Looking at the screenshots, these are popping up as plain input fields.
>
> What the. Trying to reproduce here.

This was a bug in my local database setup. Sorry for the noise.

As far as I'm concerned, you're welcome to merge now. If you want to go ahead and implement (some of) the ideas below in this branch, cool; but I'm also fine with holding off on them.

- I pull new API server migrations and haven't run them yet.
- I pull new API server test fixtures and haven't loaded them yet.

> These things need to happen around the same time "bundle install" needs to happen. Should Workbench (via run_test_servers.py or otherwise) run apiserver's "bundle install", too? (I've run into this one before, but I figured it was my own fault for not updating, not Workbench's test suite's fault for not knowing how to update.)

I have also run into this and agree that it would be nice to handle automatically.

- I run another test suite that modifies the test database and leaves it dirty. (I know making those reset-aware is on the todo list, but still.)

> Hm, a "reset to fixtures" call would address this (without being slow). We already do it before every subsequent test case, after all, so it seems like we should do it before the first, too.

Yeah, in retrospect I think this makes more sense as something to do before a test rather than afterward. If you think of it as "Ensure the necessary fixtures are in place," it sounds more like a setup task than a teardown task (the "clean up after myself" wording).

> How about this: [...] With this sort of pattern, run-tests.sh can set ARVADOS_TEST_API_INSTALLED (to a path), and Workbench will save some time. Next ARVADOS_TEST_API_HOST can point to its ip:port, and Workbench/run_test_server.py can skip their "start up an API server" stuff.
>
> And eventually everything will pay attention to the ip:port and we'll be able to run two test suites on a single host without port number catastrophe.

This all sounds like a good migration plan to me, yeah. Thanks.

#26 - 01/19/2015 04:32 PM - Tim Pierce

Reviewing 3021-api-performance at 1974e0e33.

> We use the Oj and multi_json gems, which makes Oj the default JSON parser. However, Rails' ActiveRecord::Base overrides this and uses the native JSON parser, which is slow.

Let's add a note that this should be fixed as of Rails 4.1.0 (according to the merge commit), so we can try backing out this change then and see if we still have a performance hit.

There are several other render json: calls in the application controllers:

```
hitchcock:/home/twp/arvados/services/api% git grep 'render json'
app/controllers/application_controller.rb:    render json: err, status: status
app/controllers/arvados/v1/collections_controller.rb:      render json: @object
app/controllers/arvados/v1/collections_controller.rb:    render json: visited
app/controllers/arvados/v1/collections_controller.rb:    render json: visited
app/controllers/arvados/v1/groups_controller.rb:    render json: @object_list
app/controllers/arvados/v1/keep_disks_controller.rb:      render json: @object.as_api_response(:superuser)
app/controllers/arvados/v1/nodes_controller.rb:        render json: @object.as_api_response(:superuser)
app/controllers/arvados/v1/repositories_controller.rb:    render json: {
app/controllers/arvados/v1/schema_controller.rb:    render json: discovery
app/controllers/arvados/v1/users_controller.rb:    render json: { kind: "arvados#HashList", items: @response.a
s_api_response(nil) }
app/controllers/arvados/v1/virtual_machines_controller.rb:    render json: { kind: "arvados#HashList", items:
@response }
app/controllers/database_controller.rb:    render json: {success: true}
```

If these calls shouldn't be updated the same way for some reason, we need to document what that reason is. We should have them all use the same JSON rendering method so future code archaeologists don't get baffled about the difference.

For calculating etag: I'm concerned about the possibility that log_start_state will end up saving references to the old attributes, so that after the object is changed, @old_attributes will actually point to a structure that includes some or all of the new attributes, and etag(@old_attributes) will end up returning the wrong etag. Our log unit tests seem to exercise some of this behavior, so it's probably not a serious risk, but I'm double-checking to see if there's anything else we need to hammer down there.

#27 - 01/20/2015 08:08 AM - Tom Clegg

Tim Pierce wrote:

> Let's add a note that this should be fixed as of Rails 4.1.0 (according to the merge commit), so we can try backing out this change then and see if we still have a performance hit.

Added comment. (Also moved code into a send_json() method for less repetition.)

> There are several other render json: calls in the application controllers:
> [...]

Updated to use send_json.

> For calculating etag: I'm concerned about the possibility that log_start_state will end up saving references to the old attributes, so that after the object is changed, @old_attributes will actually point to a structure that includes some or all of the new attributes, and etag(@old_attributes) will end up returning the wrong etag. Our log unit tests seem to exercise some of this behavior, so it's probably not a serious risk, but I'm double-checking to see if there's anything else we need to hammer down there.

Well, I had figured the really important part was saving the old attributes themselves (and if the old attributes are preserved, then presumably the old etag is OK to compute later), and I didn't change how that worked. But I looked into this "unit tests seem to exercise" legend, and I didn't spot any "deep inside a hash" tests, so I added one just to make sure, and success! it failed. Fixed using Marshal.load(Marshal.dump(attributes)) in 9707b0c.

It was probably fine the way it was for updates done in API calls, since they call update_attributes with brand new hashes from the client instead of mutating existing hashes like the unit test. However, modifying objects using Rails console, code in script/*.rb, etc., would have easily produced bogus logs.

Now at 7eec2f4 with master merged.

#28 - 01/21/2015 06:14 PM - Tim Pierce

LGTM at 7eec2f4. Thanks for putting my mind at ease.

#29 - 01/26/2015 04:41 PM - Tom Clegg

- Target version changed from 2015-01-28 Sprint to Arvados Future Sprints

#30 - 02/02/2015 08:41 PM - Tom Clegg

- Target version changed from Arvados Future Sprints to 2015-02-18 sprint

#31 - 02/04/2015 07:31 PM - Peter Amstutz

A couple of gem install warnings (non fatal?)

```
Could not find arvados-0.1.20150116063758 in any of the sources
Could not find json-1.8.1.gem for installation
```

FUSE test fails:

```
======================================================================
FAIL: runTest (tests.test_mount.FuseTagsUpdateTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/peter/work/arvados/services/fuse/tests/test_mount.py", line 216, in runTest
    self.assertIn('foo_tag', os.listdir(self.mounttmp))
AssertionError: 'foo_tag' not found in []
```

Also keepproxy:

```
            ********** Running services/keepproxy tests **********

# git.curoverse.com/arvados.git/services/keepproxy
/tmp/tmp.OhwNDgQYLh/src/git.curoverse.com/arvados.git/services/keepproxy/keepproxy_test.go:120: undefined: os.
Unsetenv
FAIL    git.curoverse.com/arvados.git/services/keepproxy [build failed]

           ********** Running sdk/go/arvadosclient tests **********

# git.curoverse.com/arvados.git/sdk/go/arvadosclient
/tmp/tmp.OhwNDgQYLh/src/git.curoverse.com/arvados.git/sdk/go/arvadosclient/arvadosclient_test.go:31: undefined
: os.Unsetenv
FAIL    git.curoverse.com/arvados.git/sdk/go/arvadosclient [build failed]
```

os.Unsetenv seems to be new in Go 1.4?

```
$ go version
go version go1.3.3 linux/amd64
```

There's some implicit behavior based on environment variables in run_test_server.py that would be benefit from being spelled out in the comments, e.g.

- It always tries to reset the database based on ARVADOS_API_HOST, and only starts a new server if that fails, such as ARVADOS_API_HOST missing. Then it prints ARVADOS_API_HOST, which might have come from starting a new server, or might just be echoing the existing environment.
- "stop" does not actually stop if ARVADOS_TEST_API_HOST is in the environment?
- "my_api_host" shows up in run() and stop(), but is only meaningful when (a) starting a service where the same process will stop the service later.

#32 - 02/05/2015 09:03 AM - Tom Clegg

Peter Amstutz wrote:

> A couple of gem install warnings (non fatal?)

To minimize downloading the internet, I put "bundle install --local || bundle install" in run-tests. If the gems needed to satisfy Gemfile.lock are already sitting on your system (e.g., we kept them last time around using "bundle package") it seems kind of pointless to download the latest index from rubygems.org over and over again.

I wish it didn't print the warnings, but I don't think it would be a good idea to silence stderr entirely from the bundle install --local step. Should we just tolerate this? Add a disclaimer before/after?

> FUSE test fails:

Hm, I think I saw this one while developing this branch but I haven't seen it at/after 4470ba2. I wonder if some combination of --skip options causes/prevents this, or if there's a race in a test/setup/teardown somewhere, or...? (How repeatable is this for you?)

> Also keepproxy:
>
> os.Unsetenv seems to be new in Go 1.4?

Ah, indeed. I dealt with this by removing ARVADOS_KEEP_PROXY magic from keepclient, in favor of manipulating the KeepClient object directly in the testing code so it does what we need.

(Golang discussion about adding Unsetenv reminds us that env vars are not a good way to communicate information within a Go program. If we were communicating with external processes, Go1.3 could build a custom environment when spawning them, but it has no way to remove an env var from the current process.)

> There's some implicit behavior based on environment variables in run_test_server.py that would be benefit from being spelled out in the comments, e.g.
>
> - It always tries to reset the database based on ARVADOS_API_HOST, and only starts a new server if that fails, such as ARVADOS_API_HOST missing. Then it prints ARVADOS_API_HOST, which might have come from starting a new server, or might just be echoing the existing environment.

Documented this -- and fixed it so it only tries reset-instead-of-start on a server provided by itself or via ARVADOS_TEST_API_HOST. This wait it no longer changes behavior when a test suite messes with ARVADOS_API_HOST (which doesn't sound especially unlikely). Same change in reset().

- "stop" does not actually stop if ARVADOS_TEST_API_HOST is in the environment?
- "my_api_host" shows up in run() and stop(), but is only meaningful when (a) starting a service where the same process will stop the service later.

Added/updated docstrings to clarify.

Also fixed

- a couple of bugs that only surfaced when running Python test suites *without* run-tests.sh,
- added "kill off old servers" so one improper test suite shutdown doesn't wreck all future testing ("passenger is already running") until you manually kill off the server,
- added an imperfect "check whether port is in use", too: while testing I discovered I had been a bit too optimistic about port number collisions.

Now at b205902

#33 - 02/05/2015 09:01 PM - Peter Amstutz

Still getting the foo_tag failure. This is running the whole test suite. With --only services/fuse it passes.

```
======================================================================
FAIL: runTest (tests.test_mount.FuseTagsUpdateTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/peter/work/arvados/services/fuse/tests/test_mount.py", line 216, in runTest
    self.assertIn('foo_tag', os.listdir(self.mounttmp))
AssertionError: 'foo_tag' not found in []
```

Keepproxy test compiles now, but arvadosclient is still using os.Unsetenv:

```
            ********** Running sdk/go/arvadosclient tests **********

# git.curoverse.com/arvados.git/sdk/go/arvadosclient
/tmp/tmp.G2pETtKWpA/src/git.curoverse.com/arvados.git/sdk/go/arvadosclient/arvadosclient_test.go:31: undefined
: os.Unsetenv
FAIL    git.curoverse.com/arvados.git/sdk/go/arvadosclient [build failed]

     ********** !!!!!! sdk/go/arvadosclient tests FAILED !!!!!! **********
```

Not thrilled by non-portability of reading /proc/net/tcp but if it works you might as well leave it the way it is.

Some kind of message/explanation indicating that bundle not found warnings are harmless would be nice.

I'm a little puzzled by this logic:

```
    # Before trying to start up our own server, call stop() to avoid
    # "Phusion Passenger Standalone is already running on PID 12345".
    # We want to kill it if it's our own _or_ it's some stale
    # left-over server. But if it's been deliberately provided to us
    # by a parent process, we don't want to force-kill it. That'll
    # just wreck things for the next test suite that tries to use it.
    stop(force=('ARVADOS_TEST_API_HOST' not in os.environ))
```

If ARVADOS_TEST_API_HOST is in the environment, then we should have already tried to send the reset signal. If the sending the reset signal failed, this logic won't shut down the server, but it will still try to start a new one, resulting in a 2nd API server running on a different port, but presumably connected to the same database?

#34 - 02/06/2015 01:45 AM - Tom Clegg

Peter Amstutz wrote:

> Still getting the foo_tag failure. This is running the whole test suite. With --only services/fuse it passes.

Hm, I wonder how I can go about getting that failure myself. No luck so far (or too much luck, I guess).

It looks an awful lot like a race during fuse setup. That test case has a bunch of sleep(1) although the failure happens before all that. You know that code better -- any thoughts about why this test is undependable?

I made a few little fixes in the area but none of them seem plausible as a remedy.

> Keepproxy test compiles now, but arvadosclient is still using os.Unsetenv:

Fixed.

> Not thrilled by non-portability of reading /proc/net/tcp but if it works you might as well leave it the way it is.

One portable way to do this (listen to port 0) avoids the race condition too, but requires cooperation from passenger and nginx.

```
nginx: [emerg] invalid port in "0.0.0.0:0" of the "listen" directive in
/tmp/passenger-standalone.1s8uma6/config:81
```

Attempting to listen on the chosen port would be portable, but is also more likely to make the port unusable for TIME_WAIT, depending on whether passenger uses SO_REUSEADDR.

This isn't expected to be bullet proof -- it's just about making port collisions much more rare. When someone without /proc starts getting annoyed by port number collisions, we can revisit.

> Some kind of message/explanation indicating that bundle not found warnings are harmless would be nice.

Done.

> I'm a little puzzled by this logic:
>
> [...]
>
> If ARVADOS_TEST_API_HOST is in the environment, then we should have already tried to send the reset signal. If the sending the reset signal failed, this logic won't shut down the server, but it will still try to start a new one, resulting in a 2nd API server running on a different port, but presumably connected to the same database?

True -- if ARVADOS_TEST_API_HOST is in the environment, but reset() doesn't work on it, none of the other stuff is going to work out either. I've adjusted the section just above this, so it doesn't even bother rescuing from reset() in that case. Now, if the parent provides an unresettable API, you get the original exception from reset(), which is surely more helpful than a "passenger is already running" message.

Now at dd72a4d

#35 - 02/06/2015 05:34 PM - Tom Clegg

This looks unusual. Could this be a clue about the undependable FuseTagsUpdateTest failure?

```
runTest (tests.test_mount.FuseHomeTest) ... ok
runTest (tests.test_mount.FuseMagicTest) ... ok
runTest (tests.test_mount.FuseMountTest) ... ok
runTest (tests.test_mount.FuseNoAPITest) ... ok
runTest (tests.test_mount.FuseSharedTest) ... ok
runTest (tests.test_mount.FuseTagsTest) ... fusermount: failed to unmount /tmp/tmpbvqNrX: Device or resource b
usy
ok
runTest (tests.test_mount.FuseTagsUpdateTest) ... ok
test_sanitize_filename (tests.test_mount.FuseUnitTest) ... ok
```

#36 - 02/06/2015 06:07 PM - Tom Clegg

Turns out keepproxy and passenger seem to work fine with a recently-used port, so I changed find_available_port to "bind to port 0, close socket, and use whichever port got assigned". 7939a92

#37 - 02/06/2015 10:11 PM - Peter Amstutz

3021-leave-api-running LGTM

#38 - 02/07/2015 07:24 AM - Tom Clegg

- Status changed from In Progress to Resolved

Files

workbench-fail-1.png	57.9 KB	01/06/2015	Brett Smith