openQA Project - action #89935

t/ui/27-plugin_obs_rsync_status_details.t fails in circleCI, master branch even

2021-03-11 13:54 - okurz

Status:	Resolved	**Start date:**	2021-03-11
Priority:	Normal	**Due date:**	2021-04-30
Assignee:	mkittler	**% Done:**	0%
Category:	Concrete Bugs	**Estimated time:**	0.00 hour
Target version:	Ready		
Difficulty:			

Description

Observation

https://app.circleci.com/pipelines/github/os-autoinst/openQA/5847/workflows/ea7de855-cb7c-48ba-9ae6-4ffd63cc23a1/jobs/55071/steps

```
[07:42:03] t/ui/27-plugin_obs_rsync_status_details.t .. 42/? [07:42:03] t/ui/27-plugin_obs_rsync_s
tatus_details.t .. 43/?
#   Failed test 'Proj1 dirty status is not dirty anymore'
#   at t/ui/27-plugin_obs_rsync_status_details.t line 210.
#                   'dirty on 2021-03-11 07:42:51 +0000 (standard)'
#           matches '(?^:dirty)'

#   Failed test 'Proj1 dirty is published'
#   at t/ui/27-plugin_obs_rsync_status_details.t line 211.
#                   'dirty on 2021-03-11 07:42:51 +0000 (standard)'
#     doesn't match '(?^:published)'
[07:42:03] t/ui/27-plugin_obs_rsync_status_details.t .. 45/? [07:42:03] t/ui/27-plugin_obs_rsync_s
tatus_details.t .. 46/? [07:42:03] t/ui/27-plugin_obs_rsync_status_details.t .. 48/? [07:42:03] t/
ui/27-plugin_obs_rsync_status_details.t .. 52/? [07:42:03] t/ui/27-plugin_obs_rsync_status_details
.t .. 55/? [07:42:03] t/ui/27-plugin_obs_rsync_status_details.t .. 56/?
#   Failed test 'Proj2::appliances dirty status is not dirty anymore'
#   at t/ui/27-plugin_obs_rsync_status_details.t line 210.
#                   'dirty on 2021-03-11 07:43:40 +0000 (appliances)'
#           matches '(?^:dirty)'

#   Failed test 'Proj2::appliances dirty is published'
#   at t/ui/27-plugin_obs_rsync_status_details.t line 211.
#                   'dirty on 2021-03-11 07:43:40 +0000 (appliances)'
#     doesn't match '(?^:published)'
[07:42:03] t/ui/27-plugin_obs_rsync_status_details.t .. 58/? [07:42:03] t/ui/27-plugin_obs_rsync_s
tatus_details.t .. 59/? # Looks like you failed 4 tests of 61.
                                                            [07:42:03] t/ui/27-plugin_obs_rsync_
status_details.t .. Dubious, test returned 4 (wstat 1024, 0x400)
Failed 4/61 subtests
[07:44:45] t/ui/27-plugin_obs_rsync.t ................. ok    10107 ms ( 0.04 usr  0.00 sys +  9.5
0 cusr  0.43 csys =  9.97 CPU)
[07:44:55] t/ui/28-keys_to_render_as_links.t .......... ok    22225 ms ( 0.05 usr  0.00 sys + 18.8
0 cusr  0.89 csys = 19.74 CPU)
[07:45:17]

Test Summary Report
-------------------
t/ui/27-plugin_obs_rsync_status_details.t (Wstat: 1024 Tests: 61 Failed: 4)
  Failed tests:  43-44, 56-57
  Non-zero exit status: 4
```

Suggestions

- Move to "unstable" again
- Fix the real problem

- Ensure stability
- Move back out of unstable tests

Related issues:

Related to openQA Project - action #89899: Fix flaky coverage - t/ui/27-plugi...	**Resolved**		
Related to openQA Project - action #66718: unstable/flaky/sporadic test: t/ui...	**Resolved**	**2020-05-12**	**2021-02-23**

History

#1 - 2021-03-11 13:59 - cdywan

I feel like this overlaps with #89899 - history tells me someone is likely to investigate this and fix the other one or vice verca. So I think in practice kraih should probably have a look at this.

#2 - 2021-03-11 13:59 - cdywan

- Related to action #89899: Fix flaky coverage - t/ui/27-plugin_obs_rsync_status_details.t added

#3 - 2021-03-11 14:03 - okurz

Right, I forgot to link the other ticket. Thank you.

#4 - 2021-03-11 14:04 - okurz

- Related to action #66718: unstable/flaky/sporadic test: t/ui/27-plugin_obs_rsync_status_details.t added

#5 - 2021-03-11 14:04 - okurz

And there is more :) #66718

#6 - 2021-03-11 14:23 - kraih

Ran the test 20 times locally without issues. Might only be unstable on Circle CI, or the problem is very rare.

#7 - 2021-03-11 14:25 - kraih

- Assignee set to kraih

I'll take a quick look, but i'm not very hopeful that i'll be able to fix the actual issue.

#8 - 2021-03-11 14:37 - cdywan

So this seems to be the offending test code:

```
my $dirty_status = _wait_helper("tr#folder_$ident .dirtystatuscol .dirtystatus", sub { index(shift, 'dirty') =
= -1 });
    unlike($dirty_status, qr/dirty/, "$proj dirty status is not dirty anymore");
    like($dirty_status, qr/published/, "$proj dirty is published");
```

And _wait_helper runs two loops that return whatever they find after 3 * 30 iterations if I read it correctly. Maybe the wait could be improved? Maybe it's simply slow (blind guess)?

#9 - 2021-03-11 16:17 - kraih

While working on the code and making test results more consistent, i also stumbled over something else.

```
my $lastsync = _wait_helper("tr#folder_$ident .lastsync", sub { my $v = shift; !$v || $v eq 'no data' });
unlike($lastsync, qr/$dt/, "$proj last sync forgotten");
is($lastsync, '', "$proj last sync is empty");
```

The closure stops if $lastsync matches either of !$v || $v eq 'no data'. But the test fails if the value actually is no data. Even though empty string is just an intermediate result, and no data what's expected to the the final result in a followup test. Consistent tests should always result in no data.

#10 - 2021-03-11 16:59 - kraih

Opened a PR with some improvements that should stabilize the test further. https://github.com/os-autoinst/openQA/pull/3784

#11 - 2021-03-12 04:08 - openqa_review

- Due date set to 2021-03-26

Setting due date based on mean cycle time of SUSE QE Tools

#12 - 2021-03-12 10:55 - okurz

- Status changed from New to In Progress

#13 - 2021-03-15 15:30 - kraih

- Status changed from In Progress to Feedback

With the PR applied the test should be more stable now. Lets see how it goes.

#14 - 2021-03-16 11:25 - cdywan

I'm seeing this on the latest commit on master: https://github.com/os-autoinst/openQA/runs/2120871061

#15 - 2021-03-18 19:46 - okurz

https://app.circleci.com/pipelines/github/os-autoinst/openQA?branch=master is a good reference. Most recent failure from 10h ago in https://app.circleci.com/pipelines/github/os-autoinst/openQA/5957/workflows/db92d387-41d9-4311-9b5a-305f8967e70f/jobs/56169

```
[10:40:55] t/ui/27-plugin_obs_rsync_status_details.t .. 114/? Bailout called.  Further testing stopped:  Wait limit of 60 seconds exceeded for "tr#folder_BatchedProj .dirtystatuscol .dirtystatus", no change: published on 2021-03-18 10:41:31 +0000 (containers)
FAILED--Further testing stopped: Wait limit of 60 seconds exceeded for "tr#folder_BatchedProj .dirtystatuscol .dirtystatus", no change: published on 2021-03-18 10:41:31 +0000 (containers)
make[2]: *** [Makefile:182: test-unit-and-integration] Error 255
make[2]: Leaving directory '/home/squamata/project'
make[1]: *** [Makefile:177: test-with-database] Error 2
make[1]: Leaving directory '/home/squamata/project'
make: *** [Makefile:153: test-ui] Error 2

Exited with code exit status 2
CircleCI received exit code 2
```

#16 - 2021-03-19 12:04 - mkittler

- File 27-plugin_obs_rsync_status_details-2.t added

I've also seen the failure mentioned by okurz's last comment multiple times now. I've attached the full log.

#17 - 2021-03-26 10:24 - cdywan

- Due date changed from 2021-03-26 to 2021-04-01

Moving up the **due date** due to hackweek

#18 - 2021-04-01 18:57 - okurz

- Priority changed from Urgent to Normal

Marked the test as "unstable" for now with https://github.com/os-autoinst/openQA/pull/3822 (merged) so this reduces priority.

#19 - 2021-04-07 16:44 - cdywan

- Due date changed from 2021-04-01 to 2021-04-09

Are we okay with leaving this in *unstable* for now? Or do we want this ticket to be about fixing it? Either is fine by me (and also phrased openly in the description).

#20 - 2021-04-12 11:31 - okurz

this ticket is about fixing it, as the suggestion describes:

- Move to "unstable" again
- Fix the real problem
- Ensure stability
- Move back out of unstable tests

#21 - 2021-04-14 07:25 - cdywan

- Due date changed from 2021-04-09 to 2021-04-16

- Status changed from Feedback to Workable

Thanks for clarifying. Changing it to *Workable* then.

kraih Do you want to look into that further?

#22 - 2021-04-20 13:38 - kraih

I'll take another look, but i'm not very hopeful that it's an easy fix.

#23 - 2021-04-23 13:38 - cdywan

kraih wrote:

> I'll take another look, but i'm not very hopeful that it's an easy fix.

Any progress so far? Might be worth brainstorming together? Like, braindump here or in Rocket and see if somebody else has an idea.. or maybe you'll spot it yourself. This sometimes helps me in such cases.

#24 - 2021-04-27 10:07 - kraih

I ran the test probably a hundred times locally without it failing once. Also didn't see anything obvious skimming through the code unfortunately.

#25 - 2021-04-27 11:31 - mkittler

This is the most recent example I could find (PR from 8 days ago):

```
Retry 1 of 5 …
[10:20:42] t/ui/27-plugin_obs_rsync_status_details.t .. 118/? Bailout called.  Further testing stopped:  Wait
limit of 60 seconds exceeded for "tr#folder_BatchedProj .dirtystatuscol .dirtystatus", no change: published on
 2021-04-19 10:21:24 +0000 (containers)
FAILED--Further testing stopped: Wait limit of 60 seconds exceeded for "tr#folder_BatchedProj .dirtystatuscol
.dirtystatus", no change: published on 2021-04-19 10:21:24 +0000 (containers)
Retry 2 of 5 …
[10:22:49] t/ui/27-plugin_obs_rsync_status_details.t .. ok    92771 ms ( 0.23 usr  0.02 sys + 145.32 cusr  4.3
8 csys = 149.95 CPU)
[10:24:22]
All tests successful.
```

I suppose due to some changes it now bails out but it is basically still the same failure. The log file under CircleCI's artifacts tab looks like it would only contain logs for the 2nd retry and is therefore not very useful. It would be better if the further retries would append to the file instead of replacing existing contents. However, I've apparently already attached a log file previously which contains the logs from the relevant run but I also can't find any clues there.

Since the test itself is not that complicated I suspect that the code under test might even contain a bug which only happens quite rarely.

#26 - 2021-04-27 13:33 - kraih

- Assignee changed from kraih to mkittler

#27 - 2021-04-27 16:02 - cdywan

- Due date changed from 2021-04-16 to 2021-04-30

Bumping the *due date* to Friday, and I'll try and think if I have some other ideas. Or if it makes sense to leave this until we can diagnose this better and e.g. improve re-try logic.

> The log file under CircleCI's artifacts tab looks like it would only contain logs for the 2nd retry and is therefore not very useful. It would be better if the further retries would append to the file instead of replacing existing contents.

If it helps, the logic for this lives in tools/retry. But it doesn't really handle log files. So I *think* the tests are just being re-run because RETRY=5 and prove just overwrites the files.

#28 - 2021-04-29 10:24 - andriinikitin

If we are going to fix the test - following may be considered to work around the slowness of gru jobs in CI environment are:

- use gru synchronously instead of starting in background;
- split the test into smaller tests, when we verify one part at a time, then do tear down;
- review https://github.com/os-autoinst/openQA/pull/3784 , because UI doesn't always update things automatically, so page refresh may be needed. And that depends on Gru speed, which may accumulate delay in long test;
- to reproduce problem - we can insert sleep $x into gru jobs.

But I suggest just disabling the test or comment out unstable parts.
I don't think we have other tests that rely on javascript (implemented with some shortcuts) + many gru jobs, which is hard to make reliable in slow

environment.

The whole functionality covered in test is optional and it is not big deal if we release a bug there, because:

1. There are other ways to achieve those goals with which this UI is helping.
2. It is unlikely that somebody will try to change it. And if they will - they can think about proper tests.

#29 - 2021-05-03 12:12 - mkittler

Is this really about the slowness of Gru/Minion? It *could* be considering coverage analysis in combination with forking can indeed slow down tests significantly. In other tests I have been working around this by mocking Minion. We could try the same here as well. It might now help but speeding up tests is helpful anyways. (I was under the impression that this test doesn't just take long but actually sometimes just doesn't work at all, e.g. due to a sporadic bug in the code under test. Hence I'm not sure whether mocking Minion will actually help.)

Btw, when I remember correctly, this test already does refresh the page. So this shouldn't be a problem anymore. I'd also like to note that one doesn't need to insert sleeps to simulate the slowness locally. It is sufficient to simply enable coverage analysis.

#30 - 2021-05-04 07:11 - andriinikitin

The code doesn't do anything too complicated which may be unstable:

1. UI click
2. gru task added
3. minion starts job
4. job finds mock bash script
5. mock script updates magic files (the scripts use magic files to avoid coupling with DB or other systems)
6. UI expecting to see the outcome in the updated magic files after refreshing (UI is refreshed by javascript with 2 sec timeout, but the test may refresh UI as well to see the outcome) This cycle is performed for few UI controls and several projects, so it total dozen gru jobs are used. So it is just too many subsystems are involved which are lacking tools for troubleshooting - e.g. I didn't find a way to collect Gru logs on eventual failure. (In other words it is hard to control if Gru have had any eventual problems spawning the jobs). Plus the test and frameworks force own timeouts, which may affect the flow.

I don't think enabling coverage will slow down as much as enabled coverage in CI.

#31 - 2021-05-04 14:22 - mkittler

I've been setting {with_gru => 0} to simulate Minion jobs not being processed at all. When logging the current number of pending Minion jobs in _wait_for_change via $t->app->minion->jobs({states => [qw(inactive active)]})->total the number is incremented in every loop iteration. I would have expected that the number would just always be 1. It is strange that while waiting for changes we apparently enqueue even more Minion jobs.

```
# Pending jobs: 1
# Refreshing page, waiting for "tr#folder_Batch1 .obsbuilds" to change
[debug] [A1wUNvn_FEHu] GET "/admin/obs_rsync/BatchedProj%7CBatch1/obs_builds_text"
[debug] [A1wUNvn_FEHu] Routing to controller "OpenQA::Shared::Controller::Session" and action "ensure_operator
"
[debug] [A1wUNvn_FEHu] Routing to controller "OpenQA::WebAPI::Plugin::ObsRsync::Controller::Gru" and action "g
et_obs_builds_text"
[debug] [A1wUNvn_FEHu] 200 OK (0.014076s, 71.043/s)
[debug] [gFr5uU-Qhi4y] POST "/admin/obs_rsync/BatchedProj%7CBatch1/obs_builds_text"
[debug] [gFr5uU-Qhi4y] Routing to controller "OpenQA::Shared::Controller::Session" and action "ensure_operator
"
[debug] [gFr5uU-Qhi4y] Routing to controller "OpenQA::WebAPI::Plugin::ObsRsync::Controller::Gru" and action "u
pdate_obs_builds_text"
[debug] [gFr5uU-Qhi4y] 200 OK (0.007854s, 127.324/s)
ok 9 - tr\#folder_Batch1 .obsbuilds present
# Pending jobs: 2
# Refreshing page, waiting for "tr#folder_Batch1 .obsbuilds" to change
[debug] [B6XMwmX5BJcD] GET "/admin/obs_rsync/BatchedProj%7CBatch1/obs_builds_text"
[debug] [B6XMwmX5BJcD] Routing to controller "OpenQA::Shared::Controller::Session" and action "ensure_operator
"
[debug] [B6XMwmX5BJcD] Routing to controller "OpenQA::WebAPI::Plugin::ObsRsync::Controller::Gru" and action "g
et_obs_builds_text"
[debug] [B6XMwmX5BJcD] 200 OK (0.012745s, 78.462/s)
[debug] [VPlBS-o6DhRo] POST "/admin/obs_rsync/BatchedProj%7CBatch1/obs_builds_text"
[debug] [VPlBS-o6DhRo] Routing to controller "OpenQA::Shared::Controller::Session" and action "ensure_operator
"
[debug] [VPlBS-o6DhRo] Routing to controller "OpenQA::WebAPI::Plugin::ObsRsync::Controller::Gru" and action "u
pdate_obs_builds_text"
[debug] [VPlBS-o6DhRo] 200 OK (0.011157s, 89.630/s)
ok 10 - tr\#folder_Batch1 .obsbuilds present
# Pending jobs: 3
```

POST "/admin/obs_rsync/BatchedProj%7CBatch1/obs_builds_text" invokes $self->app->gru->enqueue('obs_rsync_update_builds_text', {alias => $alias}); so no surprise. The fact that we enqueue more jobs while we're waiting for jobs to finish would explain why Minion jobs might be processes

extraordinarily slow. Considering that this is missing on andriinikitin's list of not "too complicated" things the test does I assume this is also a mistake. Maybe I can solve this after all.

#32 - 2021-05-05 11:13 - andriinikitin

So the test gives gru task some time to finish, then retries whole thing.
I don't see a big problem here, at least not the one which cannot be fixed by increasing timeout: if timeout is reasonable - it will pass in first run. If timeout is too short - the test will fail without regard of retry, so presence of retry doesn't make test fail.
I am not sure how it is "not present" in the list, as these are exact actions described there. I didn't try to be fully technically precise - just was trying to draw general idea, shame that it only made things worse.
But considering level of attitude here - I think that updating ticket is not the best way to communicate - I will remove myself from the notifications and feel free to ping me directly in rocketchat if you need my input.

#33 - 2021-05-05 17:48 - okurz

- Status changed from Workable to In Progress

https://github.com/os-autoinst/openQA/pull/3882

testing stability in https://github.com/os-autoinst/openQA/pull/3888

#35 - 2021-05-11 07:21 - cdywan

okurz wrote:

> https://github.com/os-autoinst/openQA/pull/3882
>
> testing stability in https://github.com/os-autoinst/openQA/pull/3888

PR got merged.

Btw the *due date* got a bit neglected. Do we want to polish the test more? Or is it considered good enough now?

#36 - 2021-05-11 08:33 - mkittler

I'm not sure whether the PR helped, I'm running tests again: https://github.com/os-autoinst/openQA/pull/3896

EDIT: The test is still unstable:

```
[08:40:47] t/ui/27-plugin_obs_rsync_status_details.t .. 179/? Bailout called.  Further testing stopped:  Wait
limit of 60 seconds exceeded for "tr#folder_Proj2appliances .dirtystatuscol .dirtystatus", no change: publishe
d on 2021-05-11 08:42:14 +0000 (appliances)
FAILED--Further testing stopped: Wait limit of 60 seconds exceeded for "tr#folder_Proj2appliances .dirtystatus
col .dirtystatus", no change: published on 2021-05-11 08:42:14 +0000 (appliances)
make[2]: *** [Makefile:187: test-unit-and-integration] Error 25
```

> Btw the due date got a bit neglected.

It is not like I invested that much time into the ticket. Most of the time I was out of ideas anyways.

#37 - 2021-05-11 09:30 - okurz

mkittler wrote:

> I'm not sure whether the PR helped, I'm running tests again: https://github.com/os-autoinst/openQA/pull/3896
>
> EDIT: The test is still unstable:
>
> ```
> [08:40:47] t/ui/27-plugin_obs_rsync_status_details.t .. 179/? Bailout called.  Further testing stopped:  W
> ait limit of 60 seconds exceeded for "tr#folder_Proj2appliances .dirtystatuscol .dirtystatus", no change:
> published on 2021-05-11 08:42:14 +0000 (appliances)
> FAILED--Further testing stopped: Wait limit of 60 seconds exceeded for "tr#folder_Proj2appliances .dirtyst
> atuscol .dirtystatus", no change: published on 2021-05-11 08:42:14 +0000 (appliances)
> make[2]: *** [Makefile:187: test-unit-and-integration] Error 25
> ```

do you have an idea how to shorten the wait time, maybe by checking over the API or database for prerequisites before waiting on UI elements?

And I would already consider it an improvement to do more specific retries within the test module, not relying on repeating the complete test file on make level.

#38 - 2021-05-17 08:37 - mkittler

Querying via the UI doesn't add much overhead (now that the test avoids it if there is still a pending Minion job anyways). We can always increase the number of retires within the test to avoid failures (of the overall test). Of course it only works if the test isn't completely stuck at this point anyways. We already scale the timeout for the CI, though.

#39 - 2021-05-17 11:38 - okurz

mkittler wrote:

> Querying via the UI doesn't add much overhead (now that the test avoids it if there is still a pending Minion job anyways). We can always increase the number of retires within the test to avoid failures (of the overall test). Of course it only works if the test isn't completely stuck at this point anyways. We already scale the timeout for the CI, though.

uh, ok. I am not clear on what actions you plan to do now, what feedback you are looking from others or where you are stuck.

#40 - 2021-05-18 10:13 - okurz

Discussed with mkittler. https://github.com/os-autoinst/openQA/pull/3896 looks like some request just take very very long but eventually finish. So our brute-force approach is to bump up the timeout within t/ui/27-plugin_obs_rsync_status_details.t further and test again.

#41 - 2021-05-19 13:25 - okurz

pair-worked with mkittler, created https://github.com/os-autoinst/openQA/pull/3911 to delete all potentially unstable parts of the test. mkittler plans to continue based on that and add unit-tests for all now uncovered parts.

#42 - 2021-05-20 12:04 - mkittler

PR: https://github.com/os-autoinst/openQA/pull/3914

#43 - 2021-05-25 11:54 - okurz

- Status changed from In Progress to Resolved

PR merged. As we have the test removed from the "unstable" list we will see as soon as that turns out to be not stable.

Files

27-plugin_obs_rsync_status_details-2.t	246 KB	2021-03-19	mkittler