Pulp - Story #5613

As a user, I have an API based way to report and redownload (if possible) corrupted content on the file system for one repository

10/24/2019 10:17 AM - jsherril@redhat.com

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	mdellweg	**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:	3.3.0		
Platform Release:		**Tags:**	Katello
Groomed:	Yes	**Sprint:**	Sprint 70
Sprint Candidate:	Yes	**Quarter:**	

Description

Motivation

Users can experience corrupted files either due to bit rot on mechanical hard drives or mistakenly run commands. We should make it as easy as possible to correct this situation when we can.

Idea

Make a new API endpoint that lives at /pulp/api/v3/repair/ and has 1 required parameter repository which is the href to the repository that is being requested to be repaired. Initially every request will be for a single repository.

Details

1. examines all downloaded content in the latest version and checks to see if each unit matches its expected checksum
2. re-downloads any corrupted files if possible based on RemoteArtifact entries in db 3. reports any units that could not be re-downloaded as counts

API Response

```
{
 "pulp_created": "2019-07-23T08:18:12.927007Z",
 "pulp_href": "/pulp/api/v3/task-groups/59f8a786-c7d7-4e2b-ad07-701479d403c5/",
 "repository_version_href": "/pulp/api/v3/repository/<path to the repository version/",
 "repaired": [
  "/pulp/api/v3/content/file/files/c23def43-44bc-45f4-8a6f-0310285f5339/",
  "/pulp/api/v3/content/file/files/18swef43-98s1-8d71-s8u1-0310285ffiq/"
 ],
 "unrepairable": [
  "/pulp/api/v3/content/file/files/c23def43-44bc-45f4-8a6f-0310285f5339/"
 ],
}
```

Note

This is only going to fix bitrot/corrupted files. It only inspects the content contained in the latest repository version, and does not consider if that repository version is an accurate or complete representation of the remote filesystem.

Related issues:

Related to Pulp CLI - Issue #7531: As a CLI user, I can view a report and rep...	**NEW**
Has duplicate Pulp - Story #5159: As a user, I can optionally validate and re...	**CLOSED - DUPLICATE**

Associated revisions

Revision f3a31d88 - 04/07/2020 05:08 PM - mdellweg

Repair repository version

fixes #5613 https://pulp.plan.io/issues/5613

History

#1 - 12/09/2019 10:00 PM - bmbouter

- Sprint/Milestone set to 3.1.0

#2 - 01/09/2020 10:54 PM - daviddavis

- Sprint/Milestone deleted (3.1.0)

#3 - 01/16/2020 06:38 PM - bmbouter

Question 1: which endpoint?

I was wondering if this should be an action endpoint or not underneath sync, e.g. /fix/ or /repair/. I decided it should not be because:

a) we may want to offer this functionality system-wide later
b) plugin writers would have to enable that endpoint. We could enable it for all plugins without their involvement, but that weakens the practice that repo detail endpoints are plugin owned.

So if we did it system wide would it be?

```
/pulp/api/v3/repair/
/pulp/api/v3/filesystem-repair/
/pulp/api/v3/repo-repair/
/pulp/api/v3/fix/
/pulp/api/v3/filesystem-fix/
/pulp/api/v3/repo-fix/
```

Question 2: What Pulp locks need to guard this task?

I think we need to lock on the repository so that other repository operations won't make new repo versions underneath the content while it's being examined. System-wide checking (not proposed in this ticket) would be a whole different story.

Question 3: To confirm, this only fixes "downloaded content" right?

So if we implement a "continue even if 404s occur" and say two RPMs are missing from the latest repo version. This utility won't also fix that. If it's expected to also fix that then we need to integrate this deeply with sync also somehow. What can we do that is the best regarding this case?

#4 - 01/16/2020 07:02 PM - jsherril@redhat.com

Couple of comments:

Question 2: This makes sense to me

Question 3:

"So if we implement a "continue even if 404s occur" and say two RPMs are missing from the latest repo version. This utility won't also fix that. If it's expected to also fix that then we need to integrate this deeply with sync also somehow. What can we do that is the best regarding this case?"

I think this repair task not fixing is fine, since it was never downloaded in the first place. But what would fix that? A further sync? (assuming the underlying upstream repository is fixed too).

#5 - 01/16/2020 08:28 PM - bmbouter

jsherril@redhat.com wrote:

> I think this repair task not fixing is fine, since it was never downloaded in the first place. But what would fix that? A further sync? (assuming the underlying upstream repository is fixed too).

Yes a further sync would resolve this if the upstream repository was also fixed. Pulp is going to recognize it does not have the content unit during that sync.

#6 - 01/17/2020 12:45 PM - ttereshc

> Question 3:
>
> "So if we implement a "continue even if 404s occur" and say two RPMs are missing from the latest repo version. This utility won't also fix that. If it's expected to also fix that then we need to integrate this deeply with sync also somehow. What can we do that is the best regarding this case?"
>
> I think this repair task not fixing is fine, since it was never downloaded in the first place. But what would fix that? A further sync? (assuming the underlying upstream repository is fixed too).

For the upcoming sync to fix it, a sync should be always operational, with no condition to skip it.
How confident are we that every sync being operational is good enough performance-wise? If nothing changed upstream, we'll still analyse all the repodata, we'll check if every content exists in pulp and then we may perform all the checks to ensure that repo is valid (I'm not sure if we are performing this check if nothing changed in a repo). Maybe it's fast and no need to be worried.

If we want any optimization, we'll either need to introduce force_full to pulp3, or we may need to do full repair here and not only downloaded content. What do you think?

Also we potentially can have different options for the suggested endpoints:
/pulp/api/v3/repair/ - full repair = corrupted files + download missing ones
/pulp/api/v3/filesystem-repair/ - only corrupted files
/pulp/api/v3/repo-repair/ - corrupted and downloaded for a specific repo

I'm worried that having all those options adds complexity but at the same time it gives flexibility.

#7 - 01/17/2020 10:48 PM - bmbouter

ttereshc wrote:

> For the upcoming sync to fix it, a sync should be always operational, with no condition to skip it.

Agreed

> How confident are we that every sync being operational is good enough performance-wise? If nothing changed upstream, we'll still analyse all the repodata, we'll check if every content exists in pulp and then we may perform all the checks to ensure that repo is valid (I'm not sure if we are performing this check if nothing changed in a repo). Maybe it's fast and no need to be worried.

I suspect no matter how fast it is, users will want it to be faster in cases when it can be. I'm interested in us measuring the resync time in the case that nothing changed for RPM first so we can understand how much opportunity for speedup there is.

> If we want any optimization, we'll either need to introduce force_full to pulp3, or we may need to do full repair here and not only downloaded content.
> What do you think?

What I learned from pulp2 is that anytime we introduce an optomization we need a way to turn it (or all of them) off.

> Also we potentially can have different options for the suggested endpoints:
> /pulp/api/v3/repair/ - full repair = corrupted files + download missing ones
> /pulp/api/v3/filesystem-repair/ - only corrupted files
> /pulp/api/v3/repo-repair/ - corrupted and downloaded for a specific repo
>
> I'm worried that having all those options adds complexity but at the same time it gives flexibility.

This is my primary concern also.

#8 - 01/17/2020 10:57 PM - bmbouter

- Description updated

I added clarification to the body that it only inspects the filesystem and won't check the remote metadata to consider if the repository version is "complete".

Question: how will the "unable to be fixed" units be reported?

In terms of current task reporting capabilities, progress reports are only prepared to report counts. Is that acceptable? It could look like:

```
{
    "progress_reports": [{
            "code": "fixed_count",
            "done": 5,
            "message": "The count of fixed items that were fixed.",
            "state": "completed",
            "suffix": null,
            "total": 5
        },
        {
            "code": "pre_fix_corrupted_count",
            "done": 10,
            "message": "The count of content items that were corrupted prior to fixing",
            "state": "completed",
            "suffix": null,
            "total": 10
        }
    ]
}
```

#9 - 01/21/2020 02:39 PM - jsherril@redhat.com

That looks good to me!

#10 - 02/25/2020 06:02 PM - ttereshc

- Related to Story #5159: As a user, I can optionally validate and repair content at sync time added

#11 - 02/25/2020 06:12 PM - ttereshc

FWIW, in RPM plugin we have sync optimizations, so if nothing changed in a remote repo and no changes have been made to a pulp repo, then the upcoming sync will be a no-op one. I think it's ok, it's a rare case when one hits 404 for content download and then it started working with no changes.

#12 - 02/25/2020 06:15 PM - ttereshc

- Related to deleted (Story #5159: As a user, I can optionally validate and repair content at sync time)

#13 - 02/25/2020 06:16 PM - ttereshc

- Has duplicate Story #5159: As a user, I can optionally validate and repair content at sync time added

#14 - 03/25/2020 05:58 PM - jsherril@redhat.com

- Tags Katello-P1 added

- Tags deleted (Katello-P2)

#15 - 04/01/2020 06:26 PM - bmbouter

- Subject changed from Provide ability to report and redownload (if possible) corrupted content on the file system to As a user, I have an API based way to report and redownload (if possible) corrupted content on the file system for one repository

- Description updated

#16 - 04/01/2020 07:00 PM - jsherril@redhat.com

New response format looks good to me. +1

#17 - 04/01/2020 10:00 PM - bmbouter

- Description updated

Revised after concerns about users not being able to know what was and was not repairable.

#18 - 04/01/2020 10:05 PM - daviddavis

- Groomed changed from No to Yes

- Sprint Candidate changed from No to Yes

#19 - 04/01/2020 10:09 PM - daviddavis

- Sprint set to Sprint 69

Adding to current sprint since 3.3 deadline is soon.

#20 - 04/01/2020 10:16 PM - daviddavis

- Sprint/Milestone set to 3.3.0

#21 - 04/02/2020 12:06 PM - mdellweg

- Assignee set to mdellweg

#22 - 04/02/2020 07:25 PM - mdellweg

I am in the process of creating a POC with a slightly different design. As the operation is really repairing the artifacts in a repository version, i am adding a task that is triggered by detail endpoint of repository_version with the name 'repair'. To support the repair a list of repositories workflow, a second endpoint can be added to trigger a task with subtasks of the above type.

The python bindings to access the nested endpoint look like:
pulpcore.client.pulp_file.api.repositories_file_versions_api.RepositoriesFileVersionsApi.repair

#23 - 04/03/2020 10:38 AM - pulpbot

- Status changed from NEW to POST

PR: https://github.com/pulp/pulpcore/pull/628

#24 - 04/03/2020 06:13 PM - rchan

- Sprint changed from Sprint 69 to Sprint 70

#25 - 04/07/2020 08:23 PM - mdellweg

- Status changed from POST to MODIFIED

- % Done changed from 0 to 100

Applied in changeset pulpcore|f3a31d883fdefca3d2e52cee95fd161ff0572312.

#26 - 04/15/2020 09:56 PM - ttereshc

- Status changed from MODIFIED to CLOSED - CURRENTRELEASE

#27 - 05/08/2020 07:44 PM - ggainey

- Tags Katello added

- Tags deleted (Katello-P1)

#28 - 09/18/2020 09:50 PM - daviddavis

- Related to Issue #7531: As a CLI user, I can view a report and repair corrupted content added