Pulp - Story #147

As a pulp user, I would like to publish to an rsync target

02/05/2015 09:13 PM - cduryee

Status:	NEW	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:			
Platform Release:		**QA Contact:**	
Blocks Release:		**Complexity:**	
Backwards Incompatible:	No	**Smash Test:**	
Groomed:	No	**Verified:**	No
Sprint Candidate:	No	**Verification Required:**	No
Tags:		**Sprint:**	

Description

RCM and other users (like myself!) have to publish repos to rsync targets. Currently this involves a manual rsync step.

Ideally the publish operation could perform the rsync. We would need to handle both username/password auth and ssh pubkey auth. Some investigation may be needed of the best way to store such credentials securely in Pulp so they are not available for non-admin API users to view.

deliverables:

- an rsync step in the step framework via pulp/server/pulp/plugins/util/publish_step.py. We can use subprocess for the rsync call and if rsync does not exist on the system, display an error and exit. We don't need to add rsync as a requirement to the spec file.
- define standard options for a distributor that wants to use this step. All information needed for the export should live in the distributor config, similarly to the way that the certificates required for a sync live in the importer config.
- create a new distributor for whichever type of repo you want to perform this publish. For example, in pulp_rpm it would be a new distributor class & a subclass of pulp_rpm.plugins.distributors.yum.publish.BaseYumRepoPublisher that adds the rsync step
- 100% test coverage for the above
- documentation on how to use this new distributor via API calls
- release note for new feature
- optional: support in pulp-admin to enable the new distributor. This would include status updates via the status line since the percentage would be difficult to do.

History

#1 - 03/04/2015 06:49 PM - cduryee

- Tags Sprint Candidate added

#2 - 03/04/2015 06:58 PM - cduryee

One idea for credentials would be to create a directory under /etc/pulp, like /etc/pulp/credentials.d, and users would create files of the name <repo-id>.credentials. This would allow users to control the file perms.

#3 - 03/04/2015 09:14 PM - bmbouter

Assume that we do put the credentials on the filesystem.

How will any given publish know that it should publish via rsync after publishing it to the filesystem?
How will it know what the target of the rsync will be?
Does the published version stay around on the filesystem after being rsync'd?

#4 - 03/04/2015 09:24 PM - dgregor@redhat.com

bmbouter wrote:

> Assume that we do put the credentials on the filesystem.
>
> How will any given publish know that it should publish via rsync after publishing it to the filesystem?

The rsync could be a separate distributor. The user would be responsible for publishing with that distributor when they need the content rsynced.

> How will it know what the target of the rsync will be?

Configuration of the distributor

> Does the published version stay around on the filesystem after being rsync'd?

Yes (or at least configurable).

#5 - 03/04/2015 09:44 PM - bmbouter

I think having it as a separate distributor makes sense. If we are going through the effort of making a new distributor I'll suggest that we save the username/password and/or ssh key info saved right along with the other attributes.

I think this story needs to be rewritten to describe the deliverables of a new distributor.

Is the idea to introduce the distributor first on the server side by implementing it and exposing it with an API call? The API call need some thought and also the docs would need to be updated for this change.

#6 - 03/04/2015 09:48 PM - dgregor@redhat.com

bmbouter wrote:

> Is the idea to introduce the distributor first on the server side by implementing it and exposing it with an API call? The API call need some thought and also the docs would need to be updated for this change.

I believe the API call would be the same as publishing other distributors unless I'm missing something. Exposing the distributor through pulp-admin would require additional work, but that's not needed for my particular use cases.

#7 - 03/04/2015 10:30 PM - bcourt

I would recommend doing the following:

1) Create an rsync step in the step framework via pulp/server/pulp/plugins/util/publish_step.py
2) Define standard options for a distributor that wants to use this step. All information needed for the export should live in the distributor config, similarly to the way that the certificates required for a sync live in the importer config.
3) Create a new distributor for whichever type of repo you want to perform this publish. For example, in pulp_rpm it would be a new distributor class & a subclass of pulp_rpm.plugins.distributors.yum.publish.BaseYumRepoPublisher that adds the rsync step.

#8 - 03/04/2015 10:37 PM - mhrivnak

We've had requests for other storage options, like S3. We should consider a reasonably generic way to solve this problem, assuming that rsync+ssh won't be the only method of moving published content off-box. We might even support bittorrent with the same workflow.

I like the direction Barnaby is going. Whether this is a step that optionally runs at the end of a publish, or a separate distributor that reads content from some other distributor, is up for debate. The latter has a lot of appeal, but we need to think carefully about how one distributor can reference the output of another in a consistent way.

#9 - 03/04/2015 10:42 PM - cduryee

- Description updated

#10 - 03/05/2015 05:54 AM - mhrivnak

Having given this more thought, I am going to argue tomorrow (Thursday) that we should wait on this feature and do it as part of something much broader, that would likely form the basis of pulp 3.0.

In a nutshell, I think we should consider a publication (the result of a publish operation) to be a first-class object in pulp that users can interact with. They could setup a formal promotion pipeline between repos that would ensure all content in A gets promoted to B in one simple operation, and that A's most recent publication would itself be part of that promotion. Then we can also address the option of making a publication available via rsync, boto to S3, bittorrent, etc. as a front-and-center operation against a publication. I'll share more details in a separate email.

#11 - 03/06/2015 04:02 PM - mhrivnak

- Tags deleted (Sprint Candidate)

#12 - 03/06/2015 04:44 PM - dgregor@redhat.com

mhrivnak wrote:

> Having given this more thought, I am going to argue tomorrow (Thursday) that we should wait on this feature and do it as part of something much broader, that would likely form the basis of pulp 3.0.
>
> In a nutshell, I think we should consider a publication (the result of a publish operation) to be a first-class object in pulp that users can interact with. They could setup a formal promotion pipeline between repos that would ensure all content in A gets promoted to B in one simple operation, and that A's most recent publication would itself be part of that promotion. Then we can also address the option of making a publication available

via rsync, boto to S3, bittorrent, etc. as a front-and-center operation against a publication. I'll share more details in a separate email.

This pipeline idea could also play well with https://pulp.plan.io/issues/304 (directly importing exported ISOs)

#13 - 04/05/2019 04:46 PM - amacdona@redhat.com

- Tags Pulp 3 added

#14 - 04/26/2019 10:40 PM - bmbouter

- Tags deleted (Pulp 3)