Crane - Story #81

As a user, I can see a list of repositories that crane is serving

12/19/2014 09:57 PM - mhrivnak

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	High	**Due date:**	
Assignee:	dkliban@redhat.com	**% Done:**	100%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:			
Platform Release:		**Tags:**	Pulp 2
Target Release - Crane:	1.1.0	**Sprint:**	January 2015
Groomed:		**Quarter:**	
Sprint Candidate:			

Description

Currently, there is no way to query crane and see a list of all the docker repositories it is serving. Deployers want to see such a list.

Deliverables

- some way for a user to see a list of all repos served by crane, and at least a little information about the images and tags within that repo
- list should probably be delivered via a json view, and perhaps an html view
- pagination is a nice-to-have
- Document how to protect these view using Apache (Crane itself doesn't need to protect the views, it can assume that Apache is doing that for it.)

History

#1 - 01/05/2015 09:03 PM - mhrivnak

- Priority changed from Normal to High

#2 - 01/09/2015 06:19 PM - rbarlow

- Description updated

#4 - 01/12/2015 05:47 PM - cduryee

- Sprint/Milestone set to 9

#5 - 01/16/2015 09:17 PM - skarmark@redhat.com

- Status changed from NEW to ASSIGNED

- Assignee set to skarmark@redhat.com

#6 - 01/16/2015 09:19 PM - mhrivnak

Before implementing this ourselves, it's worth looking to see if there's anything here we can use: https://github.com/vpavlin/registry-face

#7 - 01/20/2015 11:55 PM - skarmark@redhat.com

- % Done changed from 0 to 30

#8 - 01/21/2015 07:44 AM - skarmark@redhat.com

I thought about a couple of viewing options for the repositories served by crane and here are the 2 options that seem feasible given the requirement and the given timeframe.

1. json view - This will be a similar json view like what we have for images and tags.
Serving URI will http://host:port/v1/repositories and the format of the json will be -

```
{'repo_registry_id1': {'images': [image_id1, image_id2, ...], 'tags': [tag1, tag2, ...], 'protected': false},
 'repo_registry_id2': {'images': [image_id1, image_id2, ...], 'tags': [tag1, tag2, ...], 'protected': false},
 ...
```

}

2. HTML view - Instead of json, we can use Flask's 'render_template' to render it in HTML instead, although I am not really sure whether we have a strong usecase for HTML view instead of json.
In any case, it is achievable and I tried it with a couple of small examples. Information in the result will be pretty much the same as above, just in the html format.

Perhaps we want to support both?

Thoughts?

#9 - 01/21/2015 07:33 PM - bmbouter

The json representation looks fine. I do think both a json representation and an HTML representation be supported.

I propose that the web handler that serves this page inspect the "Accept" HTTP header. This will allow the client to be in control and adhere to the HTTP standards about how to specify that. If "Accept" is set to "application/json" then the json should be returned, otherwise it should return an HTML view. All other values of "Accept", or if "Accept" is missing should return the HTML view. This way machines can specify "Accept" and get what they want, but the default view is human readable.

I propose that the CSS library PatternFly be used to provide some formatting assistance for the HTML. A simple, single page layout is probably the way to go. It will improve readability, be visually consistent with a lot of other Red Hat stuff, and it should abstract cross-browser compatability problems easily enough.

Normally I would say that we should do a mockup before implementing a visual design, but in this case I think trying to render the json data using PatternFly in a reasonable way without a mockup is good enough. I can help answer questions in working with PatternFly or blocker issues that come up. I haven't worked with it specifically but it is built on top of bootstrap which I've done a lot of work with.

Also when forming HTML always make sure to validate it using the W3C HTML validator.

To represent the json data with PatternFly you'll need to write a template that loops over the json data and builds the right tags that PatternFly will know how to present visually. This is all about having the right HTML entity contain the right class label. We can talk through this some if its not straightforward from the PatternFly docs. Figuring out how to get Flask to render that simpler once you know the HTML it needs to produce.

#10 - 03/17/2015 08:15 PM - dkliban@redhat.com

- Status changed from ASSIGNED to MODIFIED

- Assignee changed from skarmark@redhat.com to dkliban@redhat.com

- % Done changed from 30 to 100

https://github.com/pulp/crane/pull/26

#11 - 03/24/2015 06:27 PM - bmbouter

- Target Release - Crane set to master

#12 - 05/01/2015 05:44 PM - mhrivnak

- Target Release - Crane changed from master to 1.1.0

#13 - 06/15/2015 10:43 PM - dkliban@redhat.com

- Status changed from MODIFIED to 5

#14 - 09/14/2015 02:40 PM - dkliban@redhat.com

- Status changed from 5 to CLOSED - CURRENTRELEASE

#15 - 03/08/2018 07:09 PM - bmbouter

- Sprint set to January 2015

#16 - 03/08/2018 07:09 PM - bmbouter

- Sprint/Milestone deleted (9)

#17 - 04/15/2019 11:21 PM - bmbouter

- Tags Pulp 2 added