Google Summer of Code

PROPOSAL

Add Support for Custom Resource Definitions to the Dashboard

By

Elijah Oyekunle

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes

Table of Contents

1. Abstract

The Kubernetes dashboard previously supported Third Party Resources (TPR), but these were replaced in Kubernetes by Custom Resource Definitions (CRD). As a result, the original TPR support was removed in Dashboard, but CRD support has not been added yet.

This proposal aims at providing a generic support for Custom Resource Definitions to the dashboard, similar to the previous TPR support.

2. Background

A Kubernetes *resource* is an endpoint in the Kubernetes API that stores a collection of *API objects* of a particular kind.

Kubernetes provides means to declare custom resources, which are not available in default Kubernetes installations. These resources are particular to a particular Kubernetes installation, and they help to make Kubernetes modular and extensible.

Originally, this functionality was provided by Third Party Resources added in v1.2 and were widely adopted by the community, but there remained some issues with TPRs that necessitated a new development initiative to manage custom resources.

ThirdPartyResource (TPR) was deprecated in Kubernetes v1.7 and removed in Kubernetes v1.8, while CustomResourceDefinition (CRD) was introduced to replace it.

Custom Resource Definitions refine the concept of extensible API resources and address issues and corner cases revealed by TPRs' success in the field. A few features were remodeled or added, among them changes to the pluralization of resource names, and the ability to create non-namespaced CRDs.

Kubernetes Dashboard originally provided support for TPRs, but this was removed and since then the Dashboard has not provided support for CRDs. Work was started on CRD support in #2449, but this was cancelled because of the focus on Angular2 migration.

Right now, with the migration further along, seems a good time to pick up work on CRD support. This project proposal aims at providing a generic CRD support in the dashboard.

3. Project Implementation

This proposal aims to extend the dashboard with all basic resource management actions for CRD and CRD object management, such as *creation*, *deletion*, *editing*, *scaling*, integration with dashboard *search*, *lists*, *logs/events* and *detail* pages.

The implementation will take inspiration from the previous implementation of TPRs, and will aim for consistency with current actions and pages. The backend implementation will follow the current pattern of fetching available CRDs and CRD objects from the apiserver, and providing these as REST APIs for the frontend to consume.

The proposed implementation will be generic because CRDs created by users will have different structures. Eventually, the dashboard might have a plugin architecture which will enable users to define how their CRDs should extend the dashboard, but that is beyond the scope of this current proposal.

There will also be detailed unit and integration tests for the implementation on both frontend and backend.

The frontend will be extended as follows[1]:

Sidebar and CRD List Page

A new *Custom Resource Definition* item will be added to the sidebar menu, and the currently available CRDs will be fetched from the backend on initial page load and shown as menu items below the label (*Figure 1*). If there are no CRDs available, the label will be hidden from the sidebar.

Clicking on any of the available CRDs in the sidebar will take the users directly to the detail page for that particular CRD (*Figure 2*).

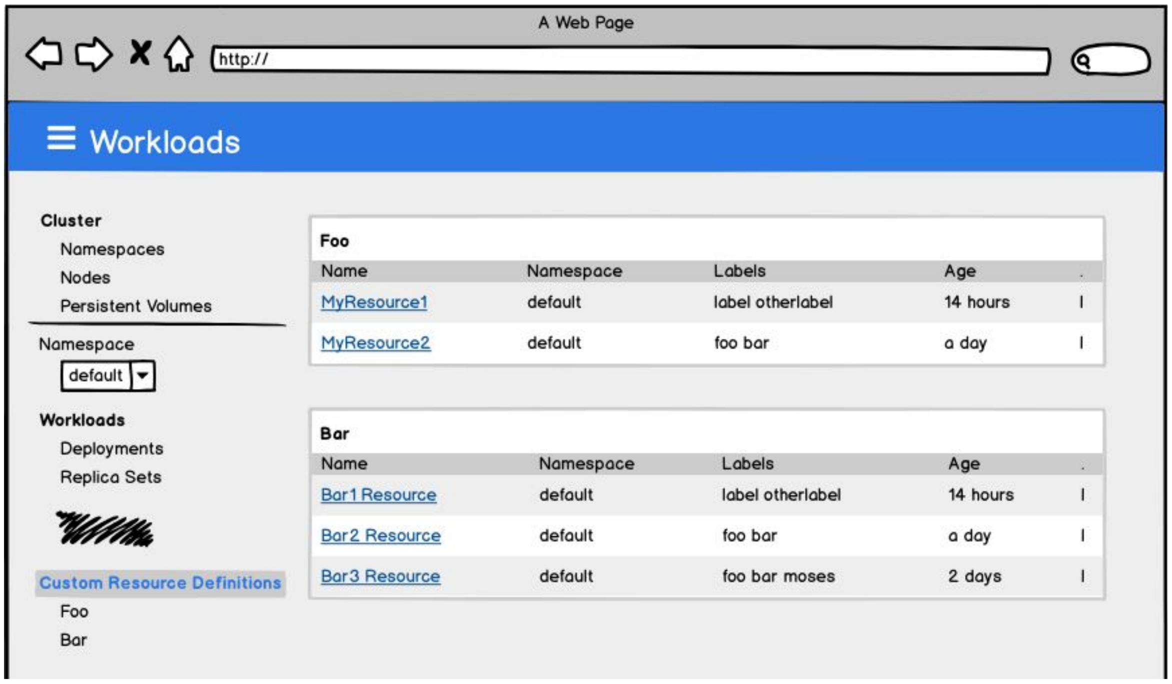
Clicking on the *Custom Resource Definition* label will take the user to a page showing all CRD objects in the cluster, grouped by CRD (*Figure 1*).

CRD Detail Page

Each CRD will be displayed on a page (*Figure 2*), showing:

- CRD Metadata

[1] Interactive wireframes for the proposed implementation are available on Balsamiq Cloud.



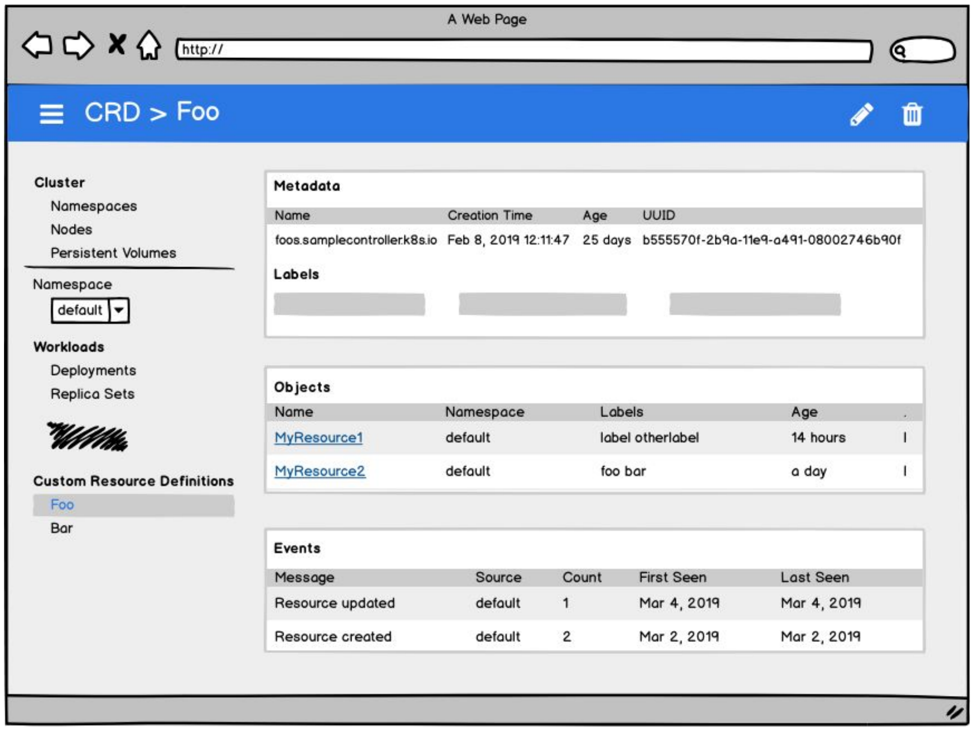
Foo

Name	Namespace	Labels	Age	.
MyResource1	default	label otherlabel	14 hours	I
MyResource2	default	foo bar	a day	I

Bar

Name	Namespace	Labels	Age	.
Bar1 Resource	default	label otherlabel	14 hours	I
Bar2 Resource	default	foo bar	a day	I
Bar3 Resource	default	foo bar moses	2 days	I

Figure 1 - CRD in Sidebar and CRD List Page



Metadata

Name	Creation Time	Age	UUID
foos.samplecontroller.k8s.io	Feb 8, 2019 12:11:47	25 days	b555570f-2b9a-11e9-a491-08002746b90f

Labels

Objects

Name	Namespace	Labels	Age	.
MyResource1	default	label otherlabel	14 hours	I
MyResource2	default	foo bar	a day	I

Events

Message	Source	Count	First Seen	Last Seen
Resource updated	default	1	Mar 4, 2019	Mar 4, 2019
Resource created	default	2	Mar 2, 2019	Mar 2, 2019

Figure 2 - CRD Detail Page

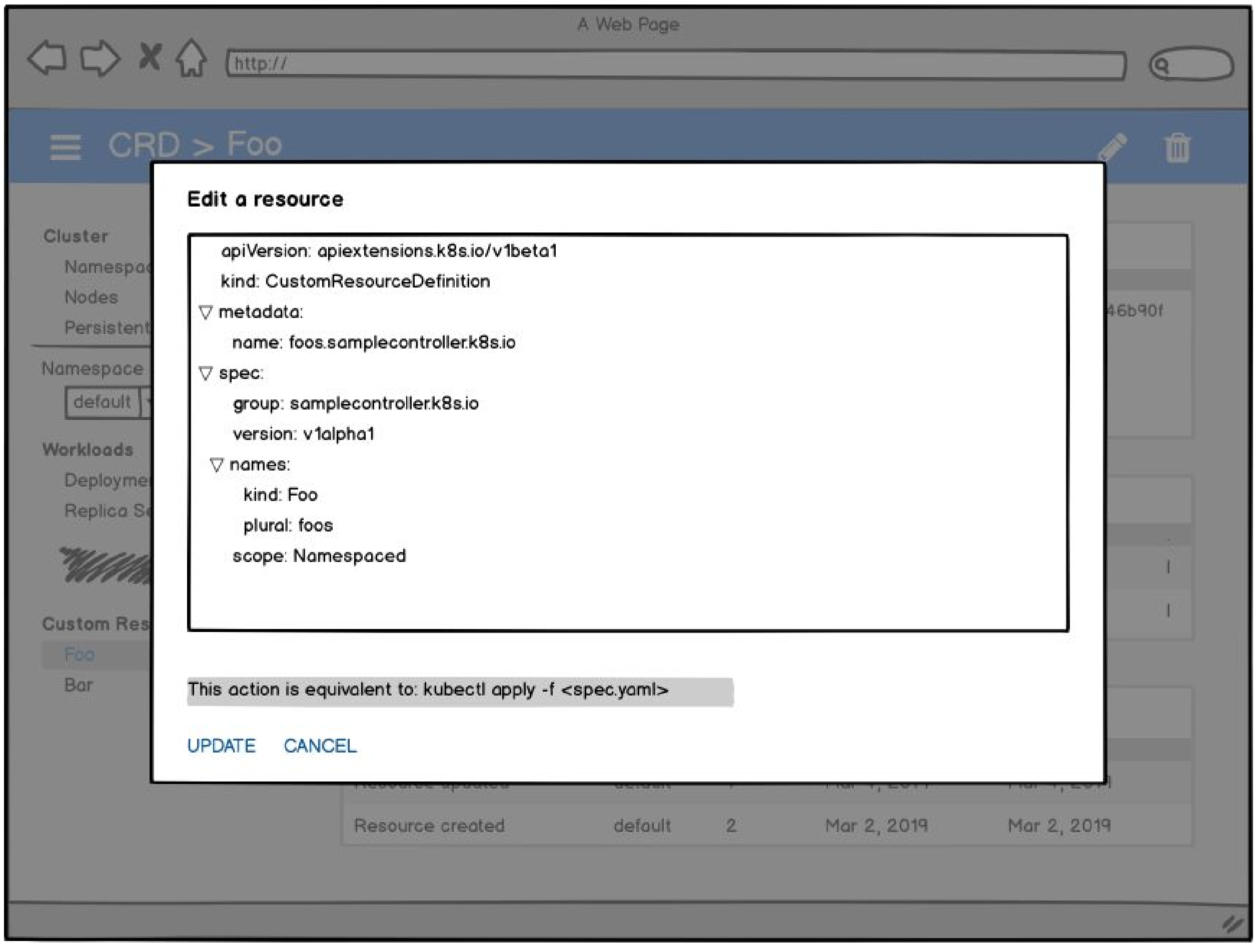
- All the CRD objects in a grid. Clicking on any of the objects will navigate to the detail page for that object (Figure 5).
- CRD event logs.

The detail page will also contain header actions to:

- *Edit* the CRD. This opens the resource in a popup as currently shown for other resources in the dashboard (Figure 3).
- *Delete* the CRD. This opens a confirmation popup (Figure 4).

Object Detail Page

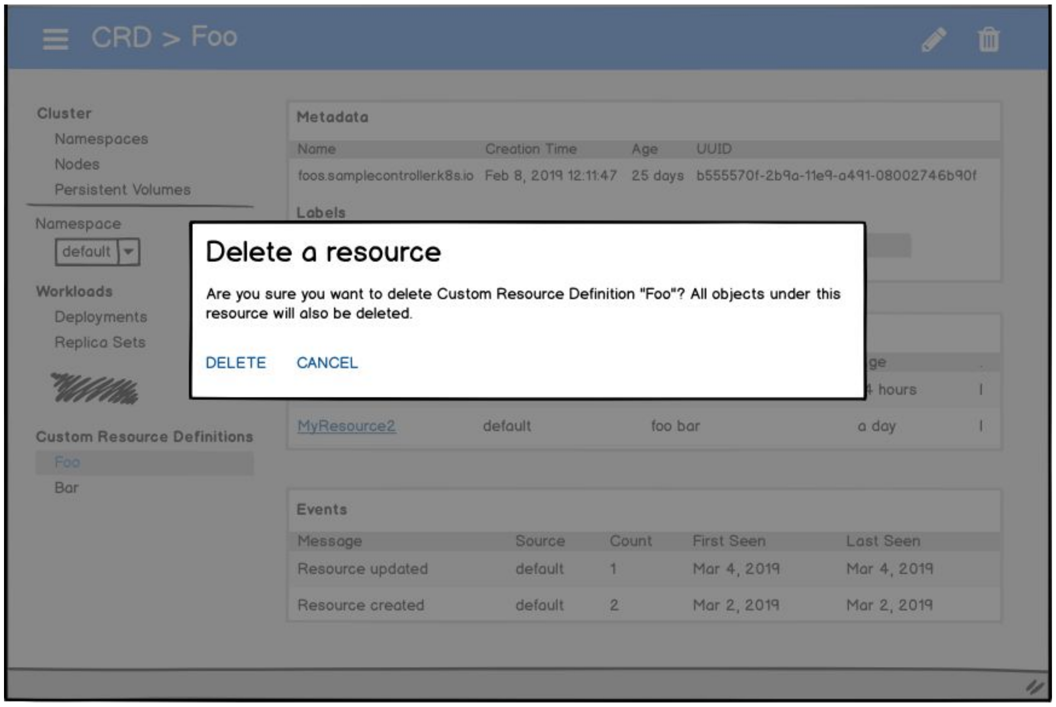
The CRD object detail page will contain metadata information and event logs for each CRD object (Figure 5).
There will be header actions on the page which display popups to *Edit* (Figure 6) and *Delete (Figure 7)* objects.



```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
▽ metadata:
    name: foos.samplecontroller.k8s.io
▽ spec:
    group: samplecontroller.k8s.io
    version: v1alpha1
  ▽ names:
      kind: Foo
      plural: foos
    scope: Namespaced
```

This action is equivalent to: kubectl apply -f <spec.yaml>

Figure 3 - Edit CRD Popup


CRD > Foo

Cluster
Namespaces
Nodes
Persistent Volumes

Namespace
default

Workloads
Deployments
Replica Sets

Custom Resource Definitions
Foo
Bar

Metadata

Name	Creation Time	Age	UUID
foos.samplecontrollerk8s.io	Feb 8, 2019 12:11:47	25 days	b555570f-2b9a-11e9-a491-08002746b90f

Labels

Delete a resource

Are you sure you want to delete Custom Resource Definition "Foo"? All objects under this resource will also be deleted.

DELETE CANCEL

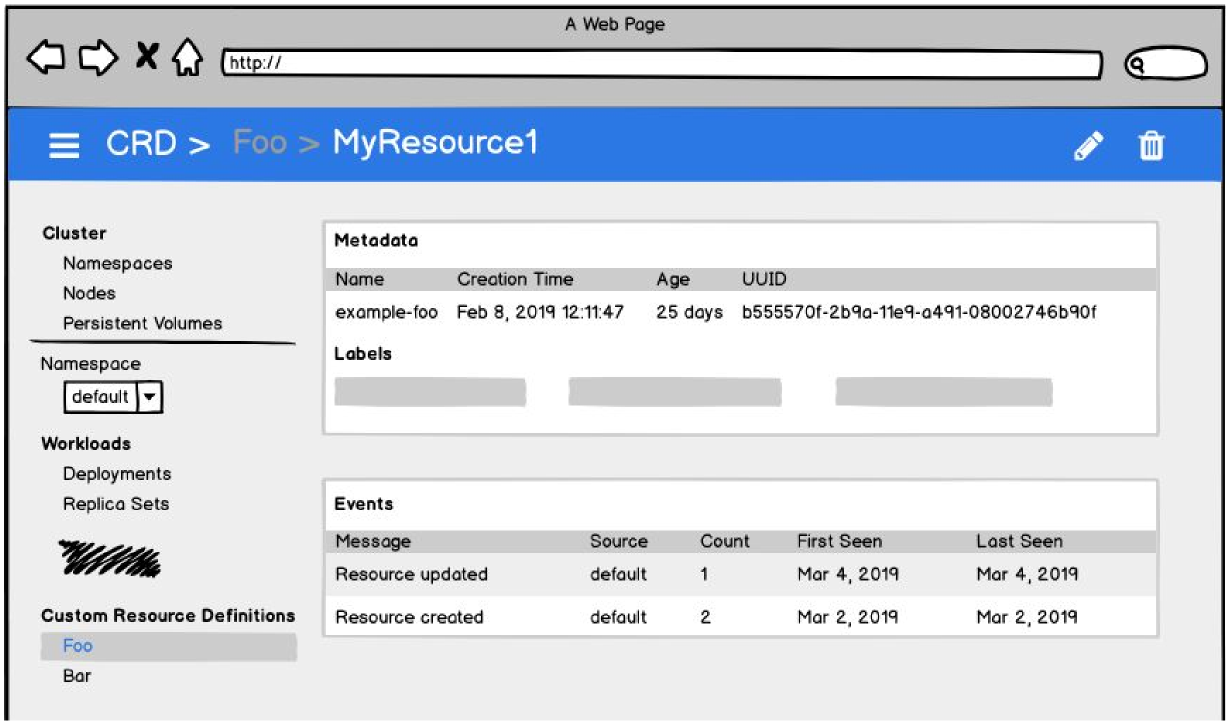
ge
4 hours

MyResource2	default	foo bar	a day	I

Events

Message	Source	Count	First Seen	Last Seen
Resource updated	default	1	Mar 4, 2019	Mar 4, 2019
Resource created	default	2	Mar 2, 2019	Mar 2, 2019

Figure 4 - Delete CRD Popup


A Web Page

http://

CRD > Foo > MyResource1

Cluster
Namespaces
Nodes
Persistent Volumes

Namespace
default

Workloads
Deployments
Replica Sets

Custom Resource Definitions
Foo
Bar

Metadata

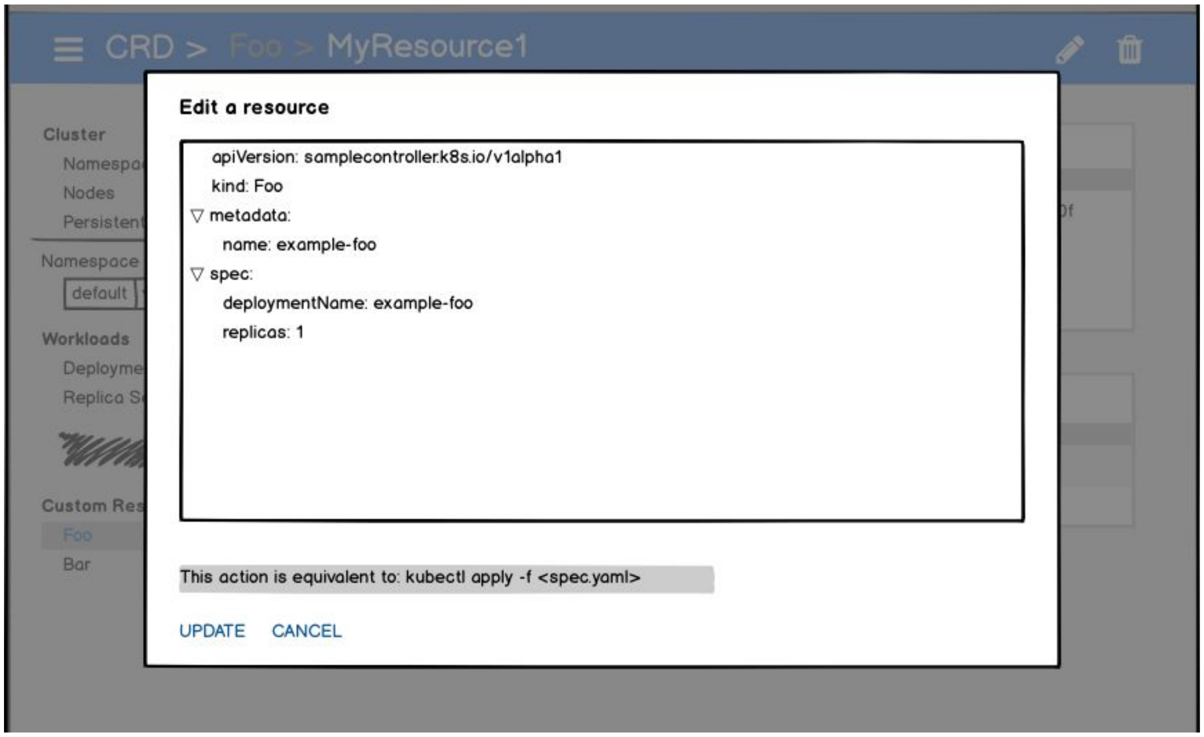
Name	Creation Time	Age	UUID
example-foo	Feb 8, 2019 12:11:47	25 days	b555570f-2b9a-11e9-a491-08002746b90f

Labels

Events

Message	Source	Count	First Seen	Last Seen
Resource updated	default	1	Mar 4, 2019	Mar 4, 2019
Resource created	default	2	Mar 2, 2019	Mar 2, 2019

Figure 5 - Object Detail Page

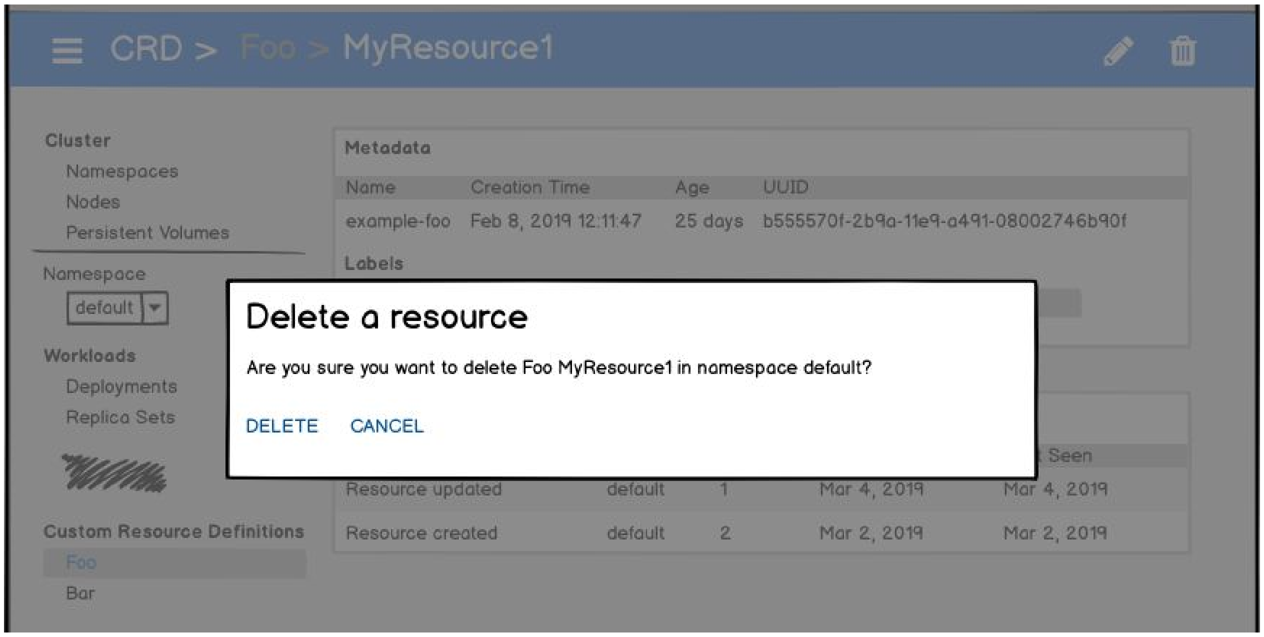


Edit a resource

```
  apiVersion: samplecontroller.k8s.io/v1alpha1
  kind: Foo
▽ metadata:
    name: example-foo
▽ spec:
    deploymentName: example-foo
    replicas: 1
```

This action is equivalent to: kubectl apply -f <spec.yaml>

Figure 6 - Object Edit Popup



Name	Creation Time	Age	UUID
example-foo	Feb 8, 2019 12:11:47	25 days	b555570f-2b9a-11e9-a491-08002746b90f

				Seen
Resource updated	default	1	Mar 4, 2019	Mar 4, 2019
Resource created	default	2	Mar 2, 2019	Mar 2, 2019

Delete a resource

Are you sure you want to delete Foo MyResource1 in namespace default?

Figure 7 - Object Delete Popup

References

- https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/ - CRD Documentation
- #2493 - Support Custom Resource Definitions - Original Github issue
- https://github.com/kubernetes/dashboard/pull/2449 - Original, cancelled CRD support.
- https://github.com/kubernetes/dashboard/blob/9a2e31afc80d60e2a282b0266897e1dd1e6b4e47/docs/design/thirdpartyresources.md - Proposal for TPRs.
- https://github.com/kubernetes/dashboard/tree/master/docs/design/mockups/24-01-2017-thirdpartyresources - Original mockups
- https://github.com/kubernetes/dashboard/pull/2491 - PR removing TPRs from Dashboard.

4. Schedule of Deliveries

The main milestones for completing this project are:

1. UI Prototypes for CRD pages.
2. Backend implementation of CRDs.
3. Frontend implementation of CRDs.
4. Fixing issues to ensure necessary integrations.

In the community bonding period, I'm planning to work with the SIG-UI Design team to come up with the looks and experience of the CRD pages, and engage in general implementation discussions.

Once we agree on looks and feel, then I plan to start working on the backend and submit for review by the community. This should be done before the second evaluation.
While the backend is being reviewed, I will start working on the frontend integration, which should be done in time for the third evaluation.

Dates	**Tasks**
May 6	Community Bonding Period begins
May 6 - May 27	Work with community on full UI/UX for CRD pages.
May 27	Community Bonding Period ends
May 27 - June 24	- Implement CRD APIs on the dashboard backend. - Add unit and integration tests for the CRDs on the backend.
June 24 - June 28	First Evaluation
June 24 - July 22	- Backend API reviewed by the community. - Initial work begins on implementing CRD support in the frontend.
July 22 - July 26	Second Evaluation
July 22 - August 19	- Implement unit tests for CRDs on the frontend. - Frontend support completed and reviewed by the community. - Submit final code and project summaries.

August 19 - August 26	Final Evaluation

Obligations

- I will be able to work full-time (a *minimum* of 40 hours a week) in May.
- I have exams in June, but I'm targeting a minimum commitment of 25hrs a week.
- I have no obligations in July and August so I'll be able to work for a minimum of 40 hours a week.

General Notes

Communication is very vital to the successful completion of GSoC, as well as coordination with the Kubernetes community. The efforts I will be making in this regard are:

- Participate in all SIG-UI bi-weekly meetings, and any other relevant SIG meetings.
- Keeping my mentor(s) informed about my progress on a daily progress.
- Weekly blog posts about project progress, including:
    - Deliverables for the week.
    - Hurdles (if any) encountered while delivering for the week.
    - How the hurdles were surmounted.
- Communicate with the community on Github and Slack.
- Maintain a public Google Doc with daily progress.
- Create a public Github tracker repository with relevant information about project progress.

5. About Me

5.1. Who am I

Name: Elijah Oyekunle

Email: eloyekunle@gmail.com

Website and Blog: https://elijahoyekunle.com

Github: https://github.com/eloyekunle

Slack (Kubernetes): eloyekunle

Twitter: https://twitter.com/elijahoyekunle

University: Department of Computer Science, The Federal University of Technology, Akure

Timezone: UTC+01:00 (West Africa Time)

5.2. Why me

I've had over 2 years of experience working with Angular2, Typescript/Javascript and Golang in live, production applications.

I have been fascinated by Kubernetes for almost 3 years, but I started using it professionally about 6 months ago. I have experience deploying applications to Kubernetes and also setting up Kubernetes clusters to cloud providers such as Azure, Google Cloud Platform and AWS. My work with Kubernetes prompted my interest in the development process behind the Kubernetes project itself.

I've started contributing to Kubernetes, and till date I've engaged in issues triage, feature requests, bug fixes and feature implementations in the Kubernetes Dashboard repository.

These are the merged pull requests I've made to the repository:

- #3586 - Fix bug in CPU allocation chart
- #3562 - Display query in search bar after namespace change and page reload
- #3561 - Logs Auto Scroll
- #3559 - Added tooltip for Timestamp icon on Logs page
- #3558 - Show light-colored scrollbar in dark logs mode

Besides these issues, I'm also currently working on dashboard charts (#3571) which are important for the version 2.0 of the dashboard. I'm also investigating a switch in the charting library we're using, evaluating possible alternatives and the possibility of maintaining a custom *d3* implementation for dashboard.

THE END