KUBERNETES AS A BATCH SCHEDULER

AUGUST 2017

AUTHOR(S):
Clenimar Souza

IT-CM-RPS

SUPERVISOR(S):
Ricardo Brito da Rocha

CERN openlab

PROJECT SPECIFICATION

Kubernetes has the notion of a Job, a higher level abstraction which adds to the underlying Pods the notions of retrials, scheduling and some error handling. This feature allows one to implement workflow engines or to use Kubernetes as a batch scheduler[1]. Also, through its Federation API, Kubernetes allows an operator to aim at multi-cluster or even multi-cloud use cases, such as cloud-bursting, hybrid cloud capabilities or high availability through globally distributed workloads[2]. The aim of this project is to: a) evaluate how much of the current CERN batch system[3] functionality, which uses HTCondor, can be replicated with Kubernetes, as well as the missing features and advantages/disadvantages; and b) evaluate the current status of Kubernetes Federations resources in comparison to single-clusters resources.

1 https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/

2 https://kubernetes.io/docs/concepts/cluster-administration/federation/

3 http://information-technology.web.cern.ch/services/batch

ABSTRACT

This project aims at executing a CERN batch use case using Kubernetes, in order to figure out what are the advantages and disadvantages, as well as the functionality that can be replicated or is missing. The reference for the batch system is the CERN Batch System, which uses HTCondor. Another goal of this project is to evaluate the current status of federated resources in Kubernetes, in comparison to the single-cluster API resources. Finally, the last goal of this project is to implement built-in support to Kubernetes federations in OpenStack Magnum, allowing operators to easily create and manage Kubernetes federations with Magnum clusters, through API calls or command line interface. OpenStack Magnum[4] is the container infrastructure component of OpenStack[5], the open-source software that allows CERN to manage its computing resources in order to create private clouds.

[4] https://wiki.openstack.org/wiki/Magnum

[5] https://www.openstack.org/software/

TABLE OF CONTENTS

1. INTRODUCTION

a. BATCH WORKLOADS: AN INTRODUCTION

In a nutshell, batch workloads are (computational) tasks to be done. It usually comprises a script or program that will perform the computation and a set of data on which the script/program will run, producing an output that is returned to the user. The whole operation is performed in a large, usually distributed computing infrastructure, which benefits large computational tasks that cannot be performed in a personal computer or workstation efficiently.

At CERN there is a service called Batch, which is used by all the experiments to run batch jobs in its more than 200.000 CPU cores, approximately 90% of the overall CERN cloud capacity. It relies on two schedulers: LSF (legacy) and more recently HTCondor. The latter is a popular scheduler in the high throughput community and has been recently integrated also with the ability to run jobs in Docker containers.

b. KUBERNETES: AN INTRODUCTION

Kubernetes is a container orchestrator engine running Docker containers under the hood. It provides a single entry point through which a user can manage all the applications and workloads running in a cluster, no matter how many nodes the cluster has. This is called "application-centric" architecture, which decouples the application from the infrastructure on which it is running: Kubernetes will take care of it and make sure that the application is always in the desired state.

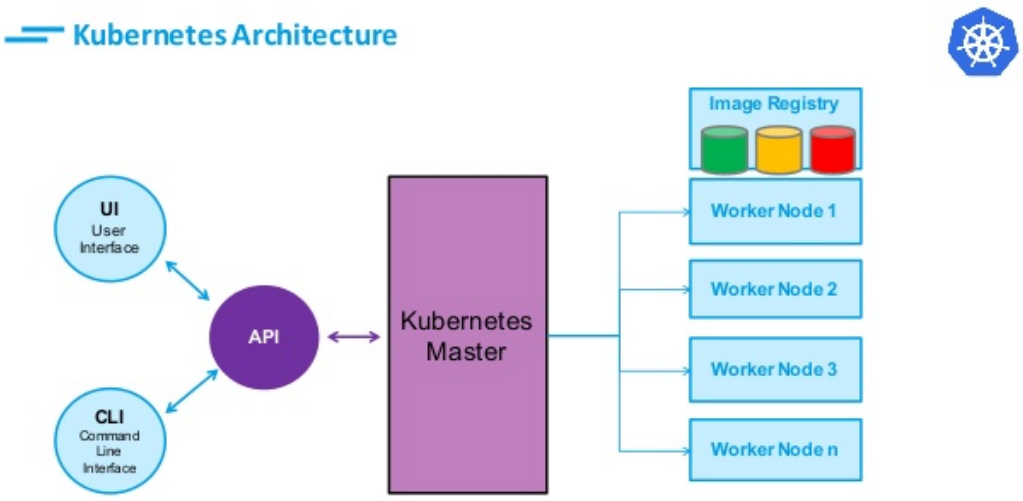

Figure 1. The Kubernetes architecture. The API component represents the single entry point for the cluster. The Kubernetes Master component will schedule the application based on the amount of resources requested by the application and the current status of the worker nodes.

c. KUBERNETES FEDERATIONS

In the case of multiple clusters, Kubernetes provides a Federation API which allows a user to easily manage a set of clusters through a single, top-level entry point. Having the ability to manage multiple clusters opens a new set of possibilities regarding the execution of applications or workloads. It's possible to hook multiple clusters in a Federation to create truly distributed applications and workloads, since the clusters might be in different availability zones, regions or even in different cloud providers. All the federated resources are created, spread and taken care by the Federation controller component, based on the amount of computing resources they require, the statuses of the clusters or personalized, user-defined scheduling policies.

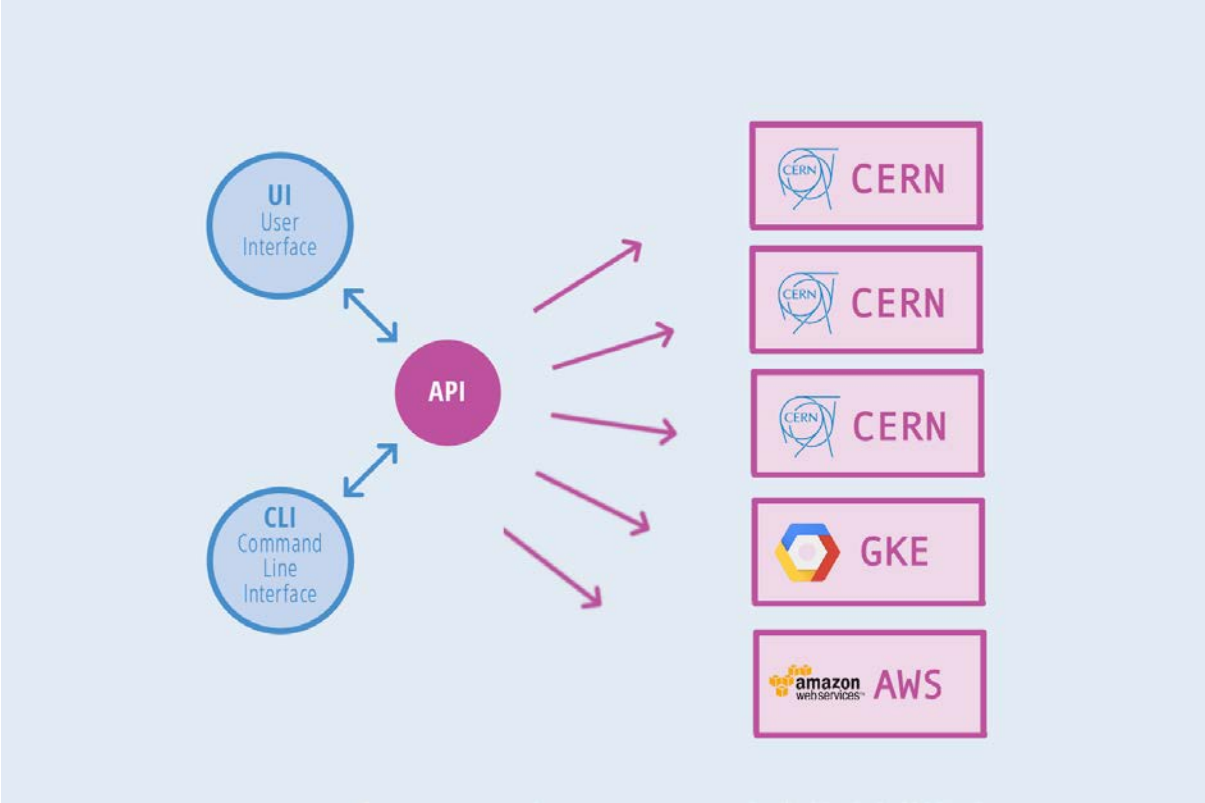

Figure 2. The Kubernetes Federation API. It is possible to create resources that will be spread to all the underlying clusters (even if they are in different cloud providers!), allowing truly distributed, high-available applications/workloads.

d. OPENSTACK MAGNUM: AN INTRODUCTION

OpenStack is the open-source software that manages most of CERN's computing resources and creates the "CERN cloud", on which all the users can run their workloads and applications. OpenStack comprises several different components, each one responsible for a specific task: Nova, for example, manages virtual machines, while Keystone provides authentication and authorization. OpenStack Magnum is responsible for container infrastructure i.e. it allows the user to create container clusters out of OpenStack resources (virtual machines) through a REST API and a command line interface.

```
$ magnum cluster-create --name mycluster \
                  --cluster-template mytemplate \
                  --node-count 8 \
                  --master-count 3
```

Figure 3. Creating a cluster with OpenStack Magnum command line interface. Magnum will talk to other OpenStack components to allocate all the resources needed.

Currently, Magnum supports three orchestrators: Kubernetes, Docker Swarm and Apache Mesos/DCOS. CERN has more than 100 Magnum clusters running in its cloud and the total number of nodes exceeds 900.

2. GOALS

This project has three main goals, and a brief summary of each one of them is given here. Later sections will approach them in depth:

1. Evaluate Kubernetes federation current status and set of functionality. Compare the status of federated resources in comparison to the regular resources. Explore multi-cluster and multi-cloud scenarios e.g. hybrid federations comprising clusters running locally at CERN and clusters running in the public cloud. Set up federations and run applications and workloads on it.
2. Run batch workloads in Kubernetes clusters. Take the CERN Batch System as reference and run a real workload in a Kubernetes cluster, evaluating how much of the Batch System functionality can be replicated in a Kubernetes cluster. Compare both approaches.
3. Add federation support to OpenStack Magnum, so a user can easily create and manage Kubernetes federations with Magnum clusters through its REST API or command line interface.

3. EVALUATE KUBERNETES FEDERATION STATUS

The first step to evaluate the current status of federations in Kubernetes is to create one federation. Since the federation was not meant to run on a public cloud provider, such as Google Cloud Platform or Amazon AWS, it was necessary to setup a DNS server first, using CoreDNS. After that, it was possible to create a Kubernetes federation running with two CERN-hosted clusters, created by Magnum. Regarding the status of the federated resources, the image below summarizes support in version 1.7.x.

Resource	Works in federation?
Deployment	Yes
Service	Yes
DaemonSet	Yes
ReplicaSet	Yes
Secret	Yes
ConfigMap	Yes
Ingress	Yes*
Job	No**

*For on-prem and hybrid setups, only nginx ingress controller.

**PR almost merged. Possibly available in the next minor releases (v1.7.x).

Figure 4. Status of Kubernetes resources in federation as of release 1.7.1.

All the steps to perform the federation setup were documented and can be found at CERN's GitLab[6].

The next step was to turn the federation into a hybrid federation, by joining a Kubernetes cluster running in the public cloud. It was possible to join Kubernetes clusters running on Google Cloud Platform, Amazon AWS, Microsoft Azure and Oracle Cloud. However, the Amazon AWS and Azure[7] clusters could not talk to the Federated API properly, causing them to be Offline (thus not being able to run applications). The other worked perfectly, and it was possible to run a distributed application across all the member clusters. The steps to join a Google Cloud Platform cluster were also documented and available at CERN's GitLab[6].

```
$ kubectl get clusters --context=fed
NAME                  STATUS    AGE
cern-c1               Ready     5d
cern-c2               Ready     5d
gke-c1                Ready     5d
oracle-c1             Ready     5d
aws-c1                Offline   5d
azure-c1              Offline   5d
```

Figure 5. All the member clusters of the federation. The federation controllers are running in cern-c1. Clusters with Ready status are able to run distributed applications.

6 https://gitlab.cern.ch/cfilemon/magnum-k8s-federation

7 https://github.com/Azure/acs-engine/issues/1118

4. RUN BATCH WORKLOADS

The initial plan was to run an actual ATLAS or CMS use case on Kubernetes to see how it performs and how much functionality it provides, but it was not possible to set up a proper container image that would allow it to be done. Instead a simple benchmark workload was run on the federation (only CERN and Google Cloud Platform clusters) to collect some CPU scores from the nodes. A sample result can be seen in the image below, where the Broadwell processor results are CERN's nodes, while the Xeon(R) processor result is the Google Cloud Platform node.

cern-federation	benchmark-bash-sa-hn-1719664166-zkqw6	Intel Core Processor (Broadwell)	2	2.9696	20.495
cern-federation	benchmark-bash-sa-hn-1719664166-9gj51	Intel Core Processor (Broadwell)	2	2.9108	20.815
cern-federation	benchmark-bash-sa-hn-3562890871-tfkqv	Intel(R) Xeon(R) CPU @ 2.30GHz	1	1.4065	21.818

Figure 6. Benchmark results showing three nodes of the federation.

Further investigations on this matter would focus on workload priorities (defining high-priority and low-priority tasks) and scheduling policies[8] (defining on which clusters that tasks must be run based on labels, annotations, selectors…).

5. ADD SUPPORT TO KUBERNETES FEDERATION TO OPENSTACK MAGNUM

OpenStack Magnum provides the means to deploy individual container clusters, supporting multiple container orchestration engines (COE) - currently Swarm, Kubernetes and Mesos/DCOS. Each of these clusters is independent from all others, with its master node(s), worker node(s) and certificate authorities from which client certificates are issued. In the case of Kubernetes, the COE has the ability to federate independent clusters, joining them under a single API endpoint and distributing workloads among all participants.

Adding built-in support to federations in Magnum would allow a set of use cases, such as managing large, heterogeneous clusters through a single endpoint with different access policies to different sets of resources, improving overall scalability.

```
$ openstack coe federation create myfederation --master cluster1
$ openstack coe federation join myfederation cluster2 cluster3
$ openstack coe federation show myfederation

+---------------------+------------------------------------------+
| Property            | Value                                    |
+---------------------+------------------------------------------+
| uuid                | 5b2ee3b5-2f85-4917-be7c-11a2c82031ad     |
| name                | myfederation                             |
| master              | <uuid-cluster1>                          |
| members             | ['<uuid-cluster2>', '<uuid-cluster3>']   |
| project_id          | ae72a4f3-30cf-4406-83b4-40b16bb480d6     |
```

[8] https://kubernetes.io/docs/tasks/federation/set-up-placement-policies-federation/

```
| properties          | dns=cluster.local                              |
| status              | UPDATE_COMPLETE                                |
| status_reason       | Federation UPDATE completed successfully       |
+---------------------+------------------------------------------------+
```

Figure 7. Example of user interactions with the OpenStack command line tool: creating a federation, joining clusters and visualizing federation info and membership.

To add such a big feature in OpenStack it is necessary to open a blueprint for the desired feature, submit a specification that defines the phases of the implementation and submit the implementation (code) itself, so the community can thoroughly review and suggest improvements in the course of the implementation.

A blueprint was created[9], as well as a specification that breaks the implementation in 5 phases:

1. Add API endpoint and data model entities;
2. Implement the federation functionality for the Kubernetes Fedora Atomic driver in Magnum (set up DNS, context, credentials…);
3. Add the new command line tools to the OpenStack client;
4. Add support for including external Kubernetes clusters (deployed in other cloud providers such as GKE, AWS, Azure);
5. Implement the Magnum federation notifications for creation, deletion and update.

The first phase was already submitted for review in two patches: DB entities[10] and API endpoints[11].

6. CONCLUSION

Kubernetes has an enormous potential and has been adopted in many companies in production environments due to its flexibility and ease to manage highly available and scalable applications. Most of the big cloud providers have created solutions that enable their users to use Kubernetes as well.

The ability to create federations and operate on multiple clusters simultaneously opens a whole set of new possibilities and interesting use cases to explore. Although under intense development, Kubernetes federations are gradually acquiring production-ready status, especially with some specific cloud providers (Google Container Engine, for instance). It was possible to explore a multi-cloud (hybrid) scenario, but we faced some problems regarding TLS configuration in two major providers (AWS and Azure), thus not being possible to attach their resources to our federation.

Running batch workloads in Kubernetes is possible and Kubernetes has a specific resource for that purpose (Job), although it is quite limited and not yet available in federation scenarios. It is possible though to use other resources such as DaemonSets or Deployments, in order to primitively simulate a task being executed. Replicating the exact same functionality set as HTCondor, however, would require more work to be done in Kubernetes, such as the ability to define priorities and retrial timeouts[12].

[9] https://blueprints.launchpad.net/magnum/+spec/federation-api

[10] https://review.openstack.org/#/c/494546/

[11] https://review.openstack.org/#/c/499193/

[12] https://github.com/kubernetes/kubernetes/pull/51153

Regarding OpenStack Magnum federation support, it was possible to implement and submit the first phase of the patch (more than 1700 lines of code!), but we still need to wait for the community to review and iterate over it. Hopefully the next phases will be pushed soon.

It was a unique experience to spend the summer as a CERN Summer Student. Working in such a brilliant work environment with such impact in the real world is very challenging, and there were a lot of work to be done. Most of it was done, but a few items were left for future work. I'd like to thank all the group, IT-CM-RPS, for the great welcoming environment they've set for the Summer Students and how they worked for us to feel like part of the group.

7. FUTURE WORK

It was a huge amount of cool stuff to be done in such a short period of time. Naturally a couple things were left for me to tackle in the future:

- Implement the blueprint for federations in OpenStack Magnum: implement, submit and iterate over the next phases of the feature;
- Investigate further on the hybrid federation issue with AWS and Azure, i.e. join their clusters into the federation and run workloads on it;
- Investigate on scheduling policies for Kubernetes Federation in order to be able to fine-grain workload placement into the various clusters.