Active Kubernetes integration

Status

Current state: "*In Progress*"

Discussion thread:

JIRA: https://issues.apache.org/jira/browse/FLINK-9953

Email:
http://apache-flink-mailing-list-archive.1008284.n3.nabble.com/DISCUSS-Best-practice-to-run-flink-on-kubernetes-td31532.html

We get lots of inputs from Till Rohrmann, Jin Sun's original design proposal. This is a more detailed doc, including user interfaces.

Motivation

Currently cloud native architectures has been introduced to many companies in production. They use kubernetes to run deep learning, web server, etc. If we could deploy the per-job/session flink cluster on kubernetes to make it mix-run with other workloads, the cluster resource utilization will be better. Also many kubernetes users are more easier to have a taste of flink.

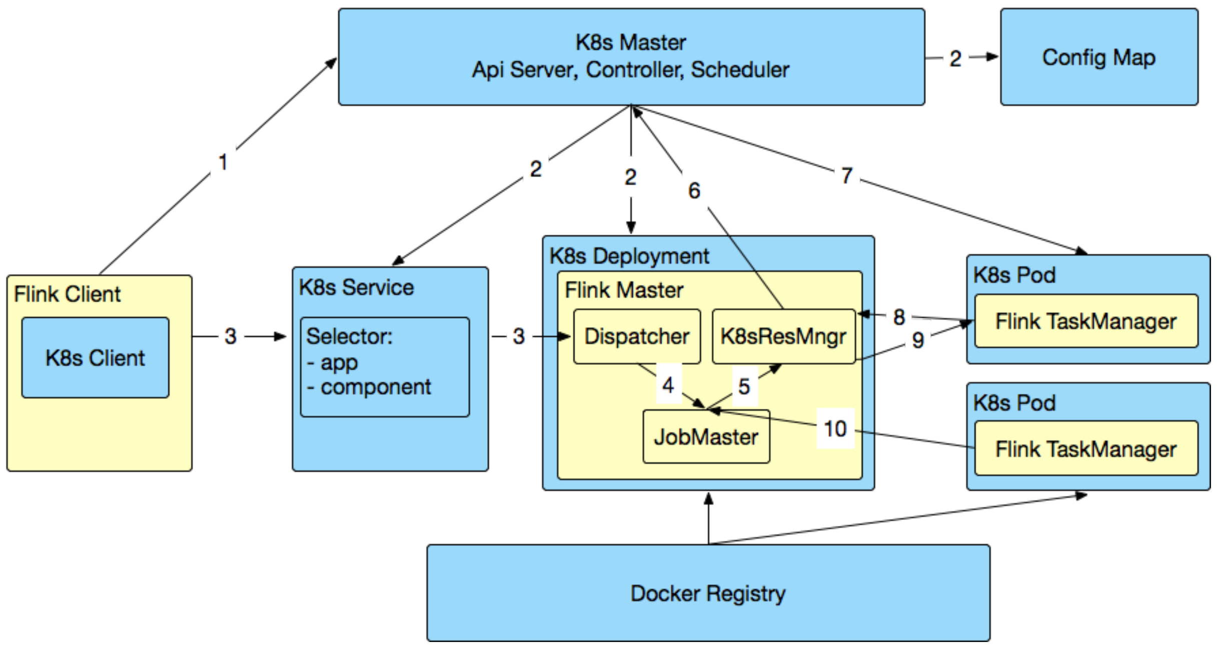
By now we have two options to run flink jobs on k8s.

- Flink standalone cluster on k8s[1]. Create jm/tm/service yaml and apply, then you will get a flink standalone cluster on k8s. Use flink run to submit job to the existed flink cluster. In order to manage multiple clusters, there are some k8s operator could be used, for example flink-k8s-operator[2]. It could manage the complete deployment lifecycle of the application. I think this option is really easy to use for the k8s users. They are familiar with k8s-operator, kubectl and other tools of k8s. They could debug and run the flink cluster just like other k8s applications.
- Natively running Flink on k8s[3]. Use the flink run or kubernetes-session.sh to start a flink cluster. It is very similar to submitting an flink cluster to Yarn. KubernetesClusterDescriptor talks to k8s api server to start a flink master deployment of 1. KubernetesResourceManager dynamically allocates resources from k8s to start task manager as demand. This option is very easy for flink users to get started. In the simplest case, we just need to update the `'-m yarn-cluster'` to -m `'-m kubernetes-cluster'`.

Architecture

This section briefly describes how Flink and Kubernetes interact.

Session Cluster



1. Flink client will first contact to Kubernetes ApiServer to submit the cluster description, including ConfigMap spec, Job Manager Service spec, Job Manager Deployment spec and Owner Reference.
2. Kubernetes Master create the flink master deployment. The Kubelet will pull the image, prepare and mount the volume and then execute the start command. After flink master pod launched, the Dispatcher and KubernetesResourceManager have been started. After step 1 and 2, the flink session cluster is ready to accept one or more jobs.
3. User submit the flink job through flink client. The job graph will be generated in flink client and then uploaded with user jars through RestClient together.
4. Once the job has been submitted successfully, JobSubmitHandler receives the request and submit job to Dispatcher. Then job master is spawned.
5. JobMaster requests slots from KubernetesResourceManager.
6. KubernetesResourceManager allocates TaskManager from k8s cluster. Each TaskManager is pod with unique id so that we could release a specific one. The KubernetesResourceManager will generate a new configuration for the TaskManagers with the address of FlinkMaster set to service name. This allows the TaskManagers to connect back to the FlinkMaster after failover.
7. TaskManager is launched.
8. TaskManager registers at SlotManager.
9. SlotManager requests slots from TaskManager.
10. TaskManager offers slots to the JobMaster. Then the tasks will be deployed and running.

PerJob Cluster

Note: User jar and dependencies are included in the user image.

Flink users could build their own image with user jar and dependencies in classpath. In this scenario, the job graph is generated on Flink Master side and we do not need to upload any jars through RestClient(step 3).

User Interfaces

Our flink on k8s users could be divided into two groups. The common users have more knowledge about flink, they could use the command line to submit job and debug job from logs of FlinkMaster and TaskManager in the kubernetes. And for platform users, they use the yaml resource files or platform web to submit flink jobs. They just want to start a flink job as simple as possible.

The option 3(native integration) will support both command line and resource files submission.

Start cluster with command cli

Note: Using the local user jar to start flink per-job cluster could not be supported. Because the user jar is located on client side, the job graph will be generated at client side and submitted to the flink master. Since in per-job mode, we use classpath job graph retriever to get the job graph. Please use kubernetes-session.sh to start an empty session or kubernetes-job.sh to start a flink per-job cluster.

Since the cli options will be deprecated and replaced with config options. So we will not provide any cli options in kubernetes mode.

Session cluster

```
./bin/kubernetes-session.sh
```

1. Start a flink session cluster

```
./bin/kubernetes-session.sh \
-Dkubernetes.cluster-id=flink-native-k8s-session-1 \
-Dkubernetes.container.image=flink:k8s \
-Djobmanager.heap.size=4096m \
-Dtaskmanager.memory.process.size=4096m \
-Dtaskmanager.numberOfTaskSlots=4 \
-Dkubernetes.jobmanager.cpu=1 -Dkubernetes.taskmanager.cpu=2 \
-Dkubernetes.container.image.pull-policy=Always
```

2. Submit a flink job to a session cluster. If the session cluster does not exist, an `Exception` will be thrown.

```
./bin/flink run -e kubernetes-session \
-Dkubernetes.cluster-id=flink-native-k8s-session-1 \
examples/batch/WordCount.jar
```

3. Attach to an existing session

```
bin/kubernetes-session.sh \
-Dkubernetes.cluster-id=flink-native-k8s-session-1 \
-Dexecution.attached=true
```

4. Delete the session cluster
In detached mode, the Flink client will exit after submitting the the service to the kubernetes cluster. If you want to stop the Kubernetes session, please use the Kubernetes utilities(`*kubectl delete service <ServiceName>*`). You can also start another client and attach to the session to stop it.

PerJob cluster

Flink user build the image with user jar and all dependencies. The jobJar is a local path in the image and in the classpath. ClassPathJobGraphRetriever will be used to build job graph. The argument '-jc' could be used to specify the entrypoint class. It will be used to generate job graph in flink master pod. If no job class is specified, it will scan classpath for job jars.

Note: Per-job cluster is still under discussion. Since in kubernetes environment, we could not ship user jars and job graph to jobmanager. Instead, users usually build their image with user jars and dependencies. So we add a new script for perjob.

```
./bin/kubernetes-job.sh
```

```
./bin/kubernetes-job.sh \
-Dkubernetes.cluster-id=flink-native-k8s-perjob-1
-Dkubernetes.container.image=flink:k8s \
-Djobmanager.heap.size=4096m \
-Dtaskmanager.memory.total-process.size=4096m \
-Dtaskmanager.numberOfTaskSlots=4
```

Start cluster with resource file

Four yaml files need to be created to start a flink on kubernetes native cluster. Compared to the standalone on kubernetes[4], there are following differences.

1. The command and args of jobmanager are different. kubernetes-entry.sh is used by docker-entrypoint.sh to start jobmanager.
2. taskmanager.yaml will not need to be created. The flink kubernetes resource manager will allocate and start taskmanager pods dynamically.

Note: This mode needs more discussion and will be supported after cli submission.

Implementation

Phase1:Basic Requirements(1.10)

A new flink-kubernetes module will be introduced with the following functionality.

- KubernetesClusterDescriptor
  - Create FlinkMaster deployment, Service
- KubernetesResourceManager
  - Support dynamic allocation/release resource for TaskManager
  - Aware namespace for resource constraint
- KubernetesSessionClusterEntryPoint
- KubernetesTaskManagerRunner
  - Set flink configuration and start TaskManager process
- FlinkKubernetesCustomCli
  - Add command options for flink-client
- KubeClient Interface and implementation
  - Interfaces to CURD k8s resources(deployment, pod, configmap, service)
  - Fabric8 Implementation
- Dockerfile, scripts to build docker images
- Documents
  - Manual for using flink command to create a session cluster

Phase2:Advanced Features(1.11)

The advanced features will make kubernetes native integration works well in production.

- Support per-job cluster
  - Running a single job Flink cluster will get better isolation. We will support cluster deploy-mode for per-job to start faster. FLINK-10934
- High-Availability
  - Use config map to store checkpoint, submitted jobgraph meta and use dfs to store the real data. Implement HighAvailabilityService based on native k8s APIs, FLINK-12884
- Multiple FlinkMasters
  - The replicas of FlinkMaster deployment could be set more than 1 to achieve a faster recovery.
- JobManager liveness check
  - The liveness of TaskManager will be controlled by Flink Master. When it failed, timeout, a new pod will be started to replace.
- Toleration

    - Set toleration for jobmanager and taskmanger.
- Node Selector
    - Deploy flink workloads to specified nodes.
- Annotation
    - Set annotations for jobmanager and taskmanager.
- LocalStorage Optimization
    - Mount emptyDir, persistentVolume, hostPath to pod to get better local storage write/read performance.
- Host Network
    - By default, we will use overlay network in the communication of all flink components. It will bring some network performance loss. So we need to support host network.
- Sider Container
    - Add a sidecar container of FlinkMaster and TaskManager to collector log to shared storage(hdfs, elastic search, etc.).
    - It could also be used for debugging purpose
- Init Container
    - Use init container to download users jars dynamically or do something else before start jobmanager and taskmanager.

References

[1].https://ci.apache.org/projects/flink/flink-docs-release-1.8/ops/deployment/kubernetes.html
[2].https://github.com/lyft/flinkk8soperator
[3].https://docs.google.com/document/d/1-jNzqGF6NfZuwVaFICoFQ5HFFXzF5NVIagUZByFMfBY/edit?usp=sharing
[4].https://ci.apache.org/projects/flink/flink-docs-release-1.9/ops/deployment/kubernetes.html