Multiple Spark/Flink clusters on YARN/Kubernetes

Pankaj Chand

How do Spark and Flink clusters get dynamically instantiated on YARN

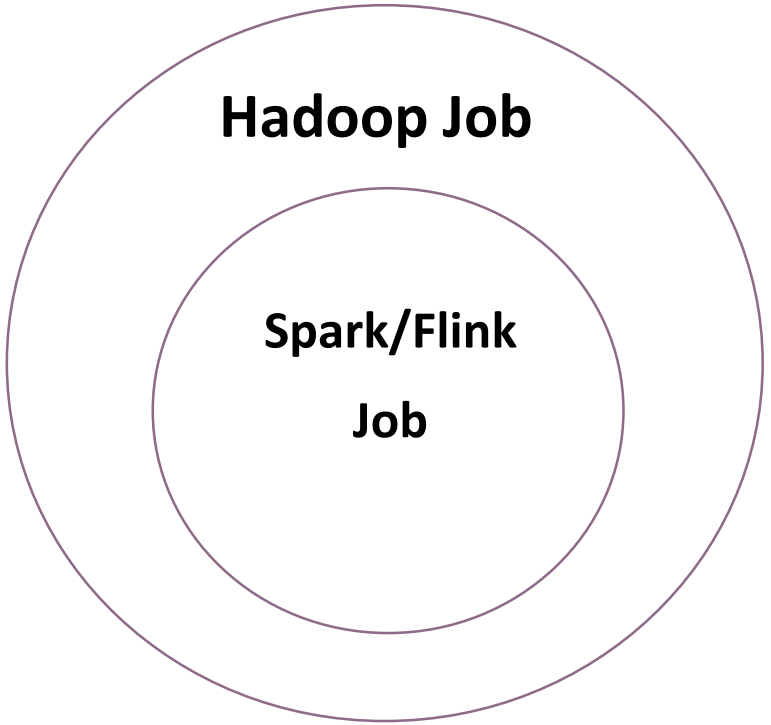
(Explanation includes details regarding Flink but the details are very similar for Spark)

Note: The latest stable versions of Spark and Flink work natively with Hadoop 2, but do not work with Hadoop 3 yet.

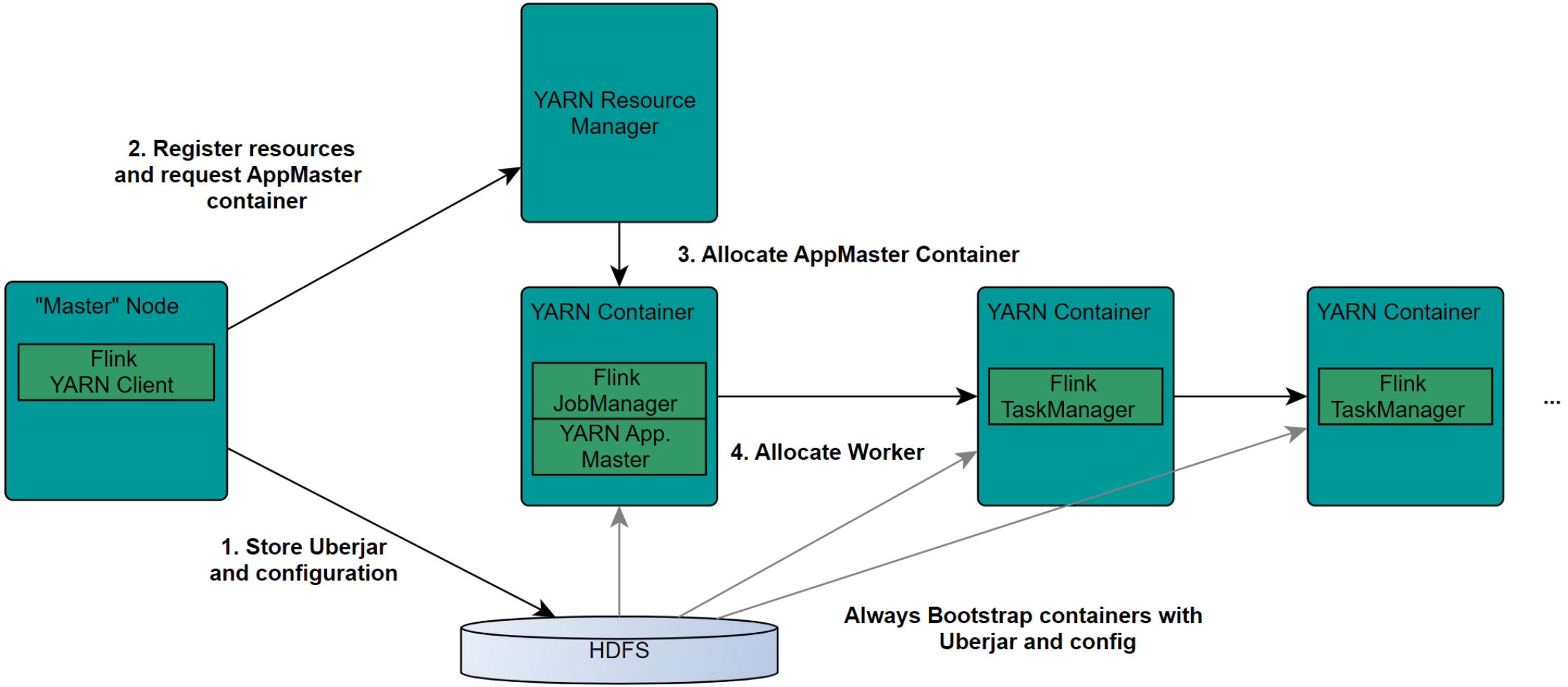
YARN creates one Application Master per application, and this Application Master is responsible for the execution of that single application. This Application Master requests for containers from the Resource Manager and executes specific programs (e.g., the main of a Java class) on the obtained containers.

Hence, YARN handles each client request for a Flink or Spark cluster by treating each such Flink/Spark cluster as a unique single application, and making one Application Master to handle each such application (Flink/Spark cluster).

- A Flink/Spark yarn-client needs to be installed on any node in the YARN cluster that is used to submit the job.
- The Flink/Spark yarn-client will submit an outer-job (i.e. Flink cluster application) and optionally, an inner-job (i.e. Flink data processing application) to the YARN cluster, and request an Application Master.
- YARN will create an Application Master and corresponding outer-job for the Flink cluster. The Flink cluster contains its own Master process and Resource Manager that will handle the resources for the inner-job.



Flink on YARN



Flink on YARN

- Flink’s Master process (JobManager) runs in the same container as the Application Master.
- All ports allocated by YARN on its containers are ephemeral ports. This allows multiple clients to execute multiple Flink and Spark sessions in parallel.
- Once the Flink cluster is ready, it either executes the inner-job (slide 4) uploaded as part of the jar-file, or waits for inner-jobs to be submitted.

Flink on Kubernetes

2 options:

First option: Run Flink, Spark, or any other system in standalone mode within containers in Kubernetes:

- Simple commands to start such a cluster (for single jobs or long-running sessions)
- Flink or Spark is unaware of Kubernetes. Hence, Flink/Spark have their own Resource Manager and Scheduler which only knows about the resources that are given to it by Kubernetes.
- Flink/Spark cannot communicate with Kubernetes' resource management and scheduling.

Flink natively on Kubernetes

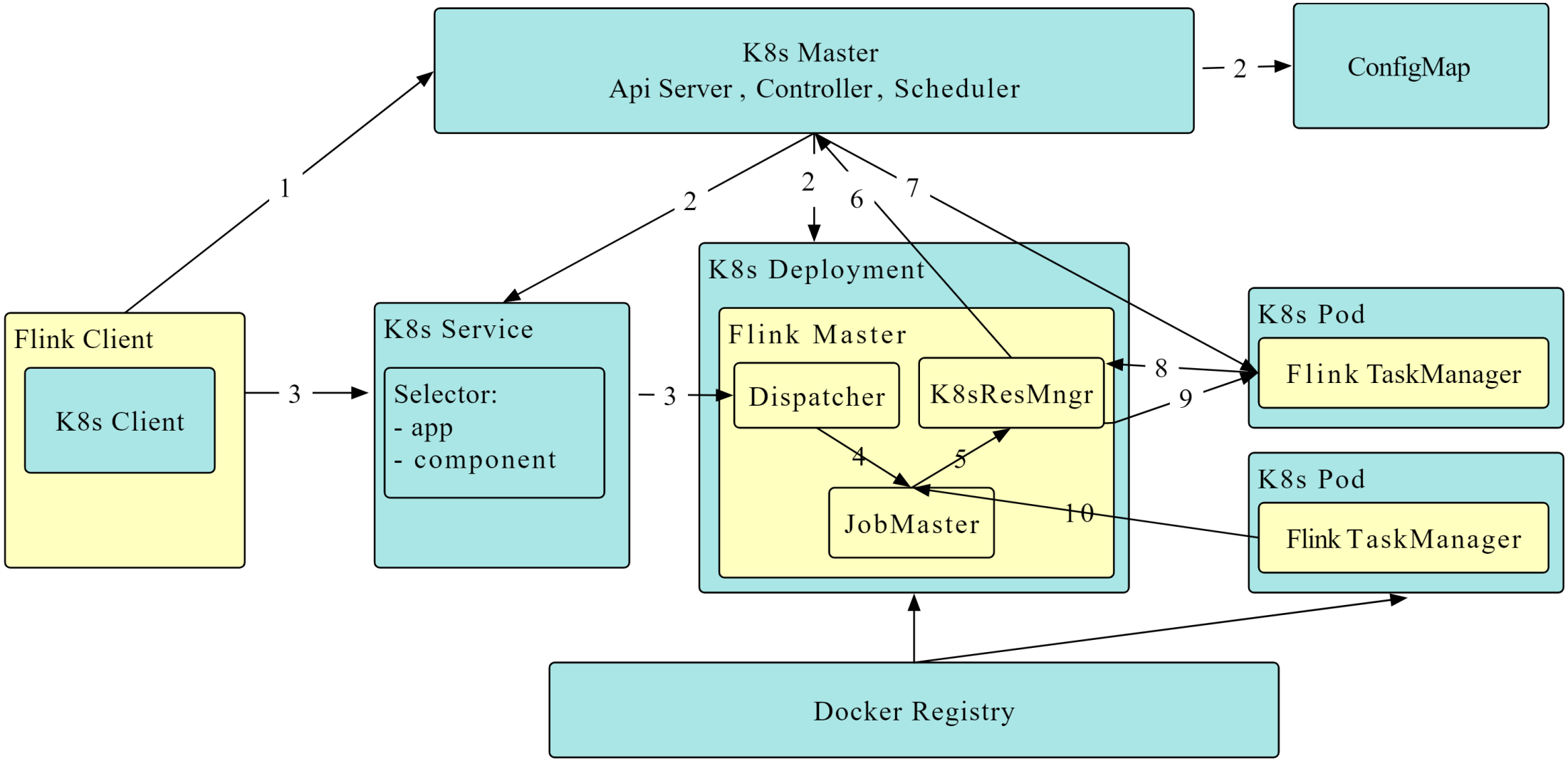
Second option: Run Flink and Spark natively in Kubernetes.

We will have a Flink/Spark kubernetes-client that will influence the Kubernetes' Resource Management and Scheduler.

Flink/Spark will communicate with Kubernetes to:

- Provision (or remove) Executors as required, i.e. Resource Elasticity
- Leverage existing Kubernetes constructs and features, such as resource quota, namespaces, logging, etc.
- Enforce multi-tenant and multi-user isolation requirements (by using Kubernetes specific features)

Flink natively on Kubernetes



Concluding points

- Flink running natively on Kubernetes is only available for long-running cluster sessions, and not for single-job clusters.
- I’m not sure if the ports generated by Kubernetes are ephemeral, but Kubernetes uses Namespaces and Resource Quotas to divide the Kubernetes-cluster resources among users. Hence, Namespaces can be used to run multiple Flink and Spark clusters in parallel.
- Running Flink natively on Kubernetes is in an experimental (Beta) stage, and so it is not complete and will be changed a lot in the near future.

Lastly, CEPH documentation states that running CEPH on Hadoop requires Hadoop 1.1.X series.

I have not seen any article where CEPH works with Hadoop 2 or Hadoop 3.