

## Flink Resource Management

Date published: 2024-06-15

Date modified: 2024-02-28



# Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Flink resource management.....</b>	<b>4</b>
Configuring the FlinkDeployment.....	5
Configuring the FlinkSessionJob.....	7

## Flink resource management

As a cluster wide resource, the CRD is the blueprint of a deployment, it defines the schema of the Flink deployments. Custom resources are the core user facing APIs of the Flink Operator that determine the Flink Application and Session cluster deployment modes.

The following operational models are supported using the Flink Deployment and Flink Session Job resources:

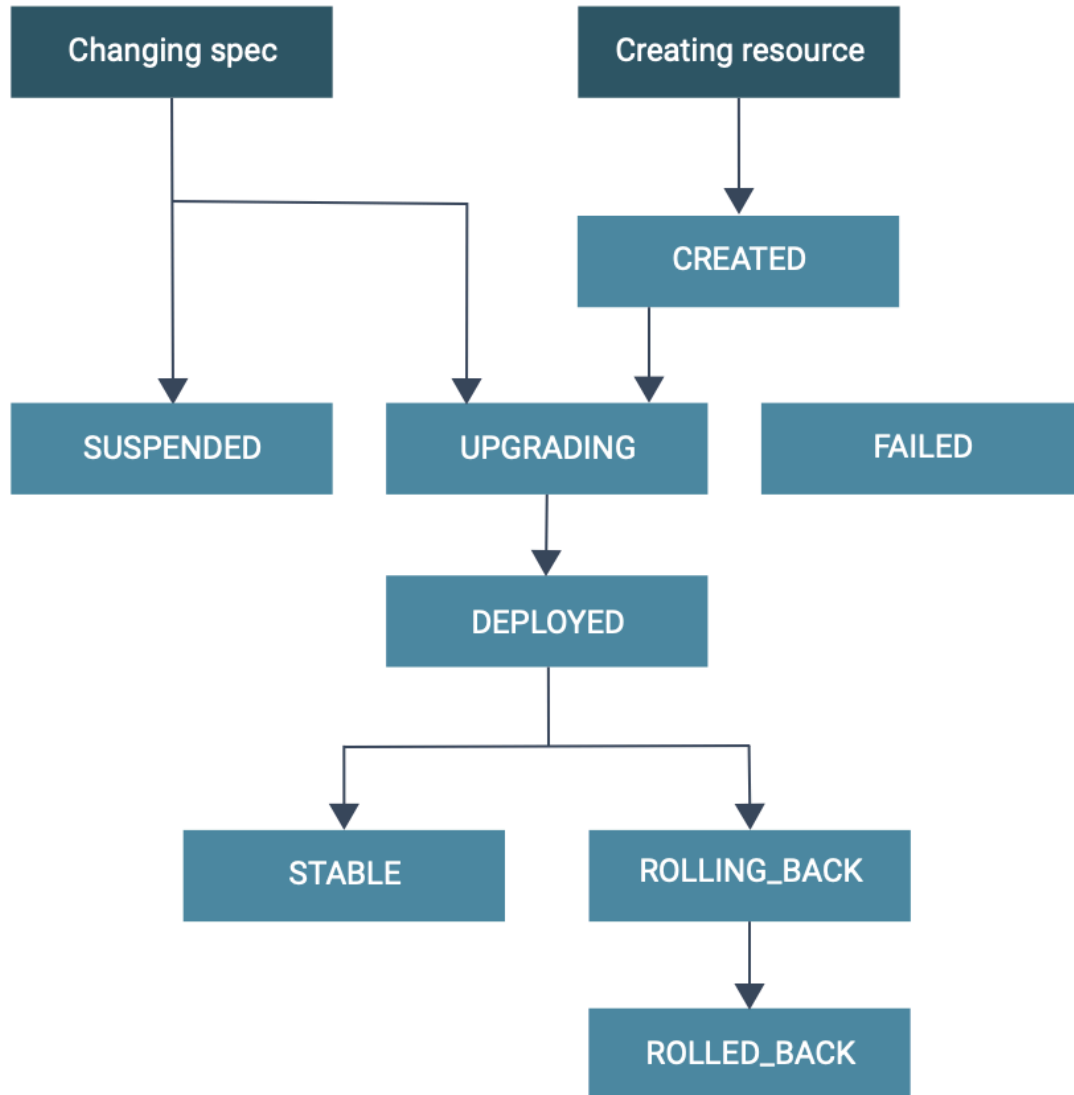
- Flink application managed by the `FlinkDeployment` resource
- Multiple jobs managed by the `FlinkSessionJob` resource sharing the same Flink "session", managed by the `FlinkDeployment` resource. The operations on the session jobs are independent of each other.

You can submit the `FlinkDeployment` and `FlinkSessionJob` configurations as YAML files using `kubectl`. The resources can be used to define, configure and manage the cluster and Flink application related resources. The Flink Operator continuously checks cluster events relating to the custom resources. When a new update is received for the custom resource, the Kubernetes cluster will be adjusted to reach the desired state. The custom resources can be applied and reapplied on the cluster anytime as the Flink Operator makes the adjustment based on the latest version.

The Flink Operator is responsible for managing the full production lifecycle of Flink resources. The following states can be identified for a Flink job resource:

- **CREATED**: The resource is created in Kubernetes, but not yet handled by the Flink Operator
- **SUSPENDED**: The Flink job resource is suspended
- **UPGRADING**: The resource is suspended before upgrading to a new spec
- **DEPLOYED**: The resource is deployed/submitted to Kubernetes, but not yet considered to be stable and might be rolled back in the future
- **STABLE**: The resource deployment is considered to be stable and will not be rolled back
- **ROLLING\_BACK**: The resource is being rolled back to the last stable spec
- **ROLLED\_BACK**: The resource is deployed with the last stable spec
- **FAILED**: The job terminally failed

The following diagram shows the transitions of the Flink application resources:



As the Flink Operator is responsible to manage the lifecycle of Flink applications, the behavior of the Flink Operator is controlled by the respective configuration property of the `spec.job` in the `FlinkDeployment` and `FlinkSessionJob` resources.

In case the `spec.job` is not specified in the `FlinkDeployment`, the Flink job will be deployed in a Session job.

## Configuring the FlinkDeployment

Learn more about the `FlinkDeployment`.

The properties of the `FlinkDeployment` resource are defined in a YAML format by the user, and must contain the following required properties:

```

apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:

```

```
namespace: namespace-of-my-deployment
name: my-deployment
spec:
  // Deployment specs of your Flink Session/Application
```

The `apiVersion` and `kind` properties have fixed values, while metadata and `spec` can be configured to control the actual Flink deployment.

The Flink Operator automatically adds status information to your `FlinkDeployment` resource based on the observed deployment state. Use the following command to check the deployment status:

```
kubectl get flinkdeployment my-deployment -o yaml
```

This returns the `FlinkDeployment` resource YAML with the current deployment status in the `jobManagerDeploymentStatus` property:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  ...
spec:
  ...
status:
  clusterInfo:
    ...
  jobManagerDeploymentStatus: READY
  jobStatus:
    ...
  reconciliationStatus:
    ...
```

To create a session cluster, you need to leave `spec.job` empty. This will result in an empty session cluster consisting only of `JobManager(s)`, after which you can use `FlinkSessionJob` resources to start jobs on these clusters.

The `spec` property is the most important part of the `FlinkDeployment` resource, as it describes the desired Flink Application or Session cluster specifications. The `spec` property contains all the information the Flink Operator needs to deploy and manage your Flink deployments, which includes Docker images, configurations, desired state and so on.

At minimum, most deployments must define the following properties:

**Table 1: Minimum configuration**

Property	Description
<code>image</code>	The Docker image of Flink containing the JAR file of the Flink job.
<code>flinkVersion</code>	Flink version used in the Docker image, for example <code>v1_19</code> .
<code>serviceAccount</code>	The Kubernetes service account of Flink that has all the permissions required to create Flink applications inside the cluster. The default value is <code>flink</code> .
<code>mode</code>	<ul style="list-style-type: none"> <li><code>native</code>: default and recommended, this integration mode uses the built-in Flink integration with Kubernetes. This means that the Flink cluster communicates directly with Kubernetes that allows Flink to manage Kubernetes resources.</li> <li><code>standalone</code>: all resources will be created at start by the Flink Operator. In this mode, Flink is unaware that it is running on Kubernetes, therefore all Kubernetes resources need to be managed externally by the Flink Kubernetes Operator.</li> </ul>
<code>taskManager</code> <code>jobManager</code>	Job and Task manager pod resource specifications (For example, CPU, memory, ephemeralStorage).

Property	Description
flinkConfiguration	List of Flink configuration overrides such as for high availability and checkpointing configurations.
job	Job specification for Application deployments. Leave it empty to create a session cluster.

## Configuring the FlinkSessionJob

Learn more about the FlinkSessionJob.

The FlinkSessionJob resource has a similar structure to the FlinkDeployment resource, as shown in the following example:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkSessionJob
metadata:
  name: basic-session-job-example
spec:
  deploymentName: basic-session-cluster
  job:
    jarURI: https://repo1.maven.org/maven2/org/apache/flink/flink-examples-streaming_2.12/1.16.1/flink-examples-streaming_2.12-1.16.1-TopSpeedWindowing.jar
    parallelism: 4
    upgradeMode: stateless
```

When using the FlinkSessionJob resource, the kind must be set to FlinkSessionJob, and the name of the target session cluster must be added to deploymentName. This target cluster must be a valid FlinkDeployment resource that you have created previously with an empty spec.job property. This will ensure that the Session deployment mode is used and the job is submitted to the session cluster.

The Flink Operator provides the following management and monitoring operations:

- Start session cluster(s)
- Monitor overall cluster health
- Stop and delete session cluster(s)

### Using pluggable file systems for FlinkSessionJob resources

The FlinkSessionJob resource can leverage both locally mounted (local storage and NFS/SAN storage) and pluggable file systems in the jarUri. In case of a pluggable file system, the Flink Operator requires the appropriate file system plugin to download the job JARs from the remote location and upload them to the running JobManager.

When a job has external JAR dependencies or requires a plugin stored on a pluggable file system, you need to extend the base flink-kubernetes-operator Docker image, as shown below, to put the related file system JAR to the plugin directory. The following example adds the Hadoop file system support JAR to the image:

```
FROM apache/flink-kubernetes-operator
ENV FLINK_PLUGINS_DIR=/opt/flink/plugins
COPY flink-hadoop-fs-1.18-SNAPSHOT.jar $FLINK_PLUGINS_DIR/hadoop-fs/
```

To add a file system plugin to the Flink Operator, you can modify the flink-kubernetes-operator.postStart property when installing the CSA Operator using Helm, which will download plugins required by the Flink Operator after it starts:

```
helm install {...} --set "flink-kubernetes-operator.postStart.exec.command={/bin/sh,-c,curl https://repo1.maven.org/maven2/org/apache/flink/flink-hadoop
```

```
-fs/1.18.1/flink-hadoop-fs-1.18.1.jar -o /opt/flink/plugins/flink-hadoop-fs-1.18.1.jar}"
```

The following pluggable files systems are supported pluggable file systems:

- Amazon S3 object storage (flink-s3-fs-presto and flink-s3-fs-hadoop)
- Aliyun Object Storage Service (flink-oss-fs-hadoop)
- Azure Data Lake Store Gen2 (flink-azure-fs-hadoop)
- Azure Blob Storage (flink-azure-fs-hadoop)
- Google Cloud Storage (gcs-connector)

For using additional types of pluggable storage, see the *File systems* page.

### Related Information

[File Systems | Apache Flink](#)