RIL Deployment

Monitoring Setup

Irtebat Shaukat

Table of Contents

Changelog

Version	Date	Changes	Drafted by by
1.0	29th May, 2025	Initial version	Irtebat
2.0	30th May, 2025	Added “Relevant Metrics” Section	Irtebat
3.0	2nd June, 2025	Added “Setup Prometheus Grafana” Section	Irtebat

Abstract

This document provides a comprehensive guide to setting up observability and monitoring for Apache Flink applications deployed using the Flink Kubernetes Operator. It outlines key metrics exposed by the Operator, JobManager, and TaskManager pods that are critical for understanding job health, state management, and performance characteristics. The document details the required configurations to enable metric scraping using Prometheus, along with the setup of visualization dashboards in Grafana. Additionally, it covers integration with Flink's REST API for enhanced monitoring and troubleshooting. The provided steps aim to equip engineering teams with a reliable monitoring stack to support production-grade Flink deployments.

Relevant Metrics

Emitted by Operator Pod

Lifecycle states

The number of resources and time spent in each of these states at any given time is tracked by the *flink_k8soperator_FlinkDeployment_Lifecycle_State_<STATE>.TimeSeconds_count* and *flink_k8soperator_FlinkDeployment_Lifecycle_State_<STATE>.TimeSeconds* metrics.

Summary of the STATES:

- CREATED : The resource was created in Kubernetes but not yet handled by the operator
- SUSPENDED : The (job) resource has been suspended
- UPGRADING : The resource is suspended before upgrading to a new spec
- DEPLOYED : The resource is deployed/submitted to Kubernetes, but it's not yet considered to be stable and might be rolled back in the future
- STABLE : The resource deployment is considered to be stable and won't 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

Selected state transitions can be tracked by the *flink_k8soperator_FlinkDeployment_Lifecycle_Transition_<Transition>_TimeSeconds_count* and *flink_k8soperator_FlinkDeployment_Lifecycle_Transition_<Transition>_TimeSeconds* metrics

Summary of Transitions:

- Upgrade : End-to-end resource upgrade time from stable to stable
- Resume : Time from suspended to stable
- Suspend : Time for any suspend operation
- Stabilization : Time from deployed to stable state
- Rollback : Time from deployed to rolled_back state if the resource was rolled back

Deployment Resource Metrics:

- Resource usage across namespace:
  - flink_k8soperator_namespace_ResourceUsage_Cpu
  - flink_k8soperator_namespace_ResourceUsage_Memory
- flink_k8soperator_namespace_JmDeploymentStatus_<STATUS>_Count
  - Note: STATUS may take values from READY, DEPLOYED_NOT_READY, DEPLOYING, MISSING, ERROR

Emitted by JobManager Pods

Checkpointing and State Related

- flink_jobmanager_job_lastCheckpointDuration
- flink_jobmanager_job_lastCheckpointSize
- flink_jobmanager_job_lastCheckpointFullSize
- flink_jobmanager_job_numberOfCompletedCheckpoints
- flink_jobmanager_job_numberOfFailedCheckpoints
- flink_jobmanager_job_totalNumberOfCheckpoints
- flink_jobmanager_job_lastCompletedCheckpointId
- flink_jobmanager_job_lastCheckpointPersistedData

Job Health and Lifecycle

- flink_jobmanager_job_uptime
- flink_jobmanager_job_runningTime
- flink_jobmanager_job_failingTime
- flink_jobmanager_job_deployingTime
- flink_jobmanager_job_restartingTime
- flink_jobmanager_job_fullRestarts

Cluster Resource Coordination

- flink_jobmanager_numRegisteredTaskManagers
- flink_jobmanager_taskSlotsTotal
- flink_jobmanager_taskSlotsAvailable
- flink_jobmanager_numRunningJobs

JVM/Infrastructure

- flink_jobmanager_Status_JVM_Memory_Heap_Used
- flink_jobmanager_Status_JVM_Memory_NonHeap_Used
- flink_jobmanager_Status_JVM_CPU_Load
- flink_jobmanager_Status_JVM_GarbageCollector_All_Count
- flink_jobmanager_Status_JVM_Threads_Count

Emitted by TaskManager Pods

Record Latency Tracking

Tracks the time it takes for special *latency marker events* to travel through the job pipeline.

Important : Flink uses latency markers for tracking record latency. Latency markers do not account for the time user records spend in operators as they bypass them. The latency measured using the markers will reflect that operators are not able to accept new records (queuing)

Note: Related metrics should to be enabled via FlinkApplication CRD as:

```
Unset
  spec:
   flinkConfiguration:
     metrics.latency.interval: "1000"
```

Note: Enabling latency metrics impacts the performance of the cluster. Flink community recommends to use them in non-production environments for debugging purposes.

- *flink_taskmanager_job_latency_source_id_operator_id_operator_subtask_index_latency.*
  - Ex: To evaluate a specific operator:
    - flink_taskmanager_job_latency_source_id_operator_id_operator_subtask_index_latency{operator_id="<>"}

Key State Access Latency

This tracks latency when accessing keyed state

Note: Related metrics should to be enabled via FlinkApplication CRD as:

```
Unset
   spec:
    flinkConfiguration:
      state.backend.latency-track.keyed-state-enabled : "true"
      #state.backend.latency-track.sample-interval: "100"
      #state.backend.latency-track.history-size: "128"
```

Note: Enabling state-access-latency metrics can impact the performance of the cluster. Flink community recommends to use them in non-production environments for debugging purposes.

Refer here for available metrics:
https://nightlies.apache.org/flink/flink-docs-master/docs/ops/metrics/#state-access-latency

Performance metrics

Metric	Type	Description
flink_taskmanager_job_task_numBytesOut	Counter	Measures total bytes emitted by the task. Indicates output volume.
flink_taskmanager_job_task_numBytesOutPerSecond	Meter	Real-time throughput. Track spikes/drops in output rate.

flink_taskmanager_job_task_isBackPressured	Gauge (0 or 1)	Shows whether the task is back-pressured. Crucial for identifying downstream slow consumers.
flink_taskmanager_job_task_backPressuredTimeMsPerSecond	Gauge	Quantifies how long a task is back-pressured. Useful for identifying persistent pressure zones.
flink_taskmanager_job_task_idleTimeMsPerSecond	Meter	Helps differentiate between idle vs blocked tasks. High values may indicate data starvation or skew.
flink_taskmanager_job_task_busyTimeMsPerSecond	Gauge	Tracks actual processing time. Can be low if task is frequently idle or blocked.
flink_taskmanager_job_<>_currentInputNWatermark	Gauge	The last watermark this operator has received in its N'th input (in milliseconds), with index N starting from 1.

flink_taskmanager_job_<>_currentOutputWatermark	Gauge	The last watermark this operator has emitted (in milliseconds).

Setup Prometheus-Grafana

Install Prometheus and Grafana using Kube-promethues Helm

Add Prometheus Helm Repository

```
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts

helm repo add stable https://charts.helm.sh/stable
```

Update Helm Repositories

```
helm repo update
```

Install kube-prometheus

```
helm install prometheus prometheus-community/kube-prometheus-stack -n
monitoring
```

> Note : kube-prometheus-stack automatically starts monitoring your Kubernetes cluster with a set of default rules. These default rules include scraping many of the standard Kubernetes resources (like nodes, pods, services, etc.) as well as system components like kubelet, kube-apiserver, kube-scheduler, etc. To prevent the default cluster monitoring from starting, we are allowed to modify these values via Helm chart configuration.
>
> The values.yaml file is modular, allowing us to configure and fine-tune nearly every component in the Prometheus/Grafana stack, from basic resource limits to specific service monitoring rules. If you're deploying a custom monitoring setup, you can disable default behaviors, add your own scrape configs, or expose only the components you need.

Start Kubernetes Prometheus Port Forwarding

Using kubectl port-forward you can see the Grafana dashboard with the data from the Metrics API by opening a browser to http://localhost :3000/ and the Prometheus dashboard at http://localhost:9090/.

```
kubectl port-forward svc/prometheus-operated 9090
kubectl port-forward deployment/prometheus-grafana 3000
```

Logging in to Grafana

Browser and open localhost:3000. Since you are logging in for the first time, you will have to use a default username and password. However, you can create new users and update passwords later.

Retrieve credentials for the dashboard :

```
kubectl get secret prometheus-grafana -o jsonpath="{.data.admin-user}"
| base64 --decode ; echo
kubectl get secret prometheus-grafana -o
jsonpath="{.data.admin-password}" | base64 --decode ; echo
```

Configure observability for Flink applications

Apache Flink exposes a variety of metrics for both the Flink Kubernetes Operator and the Flink applications (JobManager and TaskManager). These metrics must be configured and scraped separately.

Prometheus Metrics Configuration for Flink Kubernetes Operator

1. Enabling Metrics in Operator via Helm Chart

`Edit the values.yaml used for deploying the Flink Kubernetes Operator:`

```
Unset
defaultConfiguration:
  create: true
  append: true
  flink-conf.yaml: |+
    kubernetes.operator.metrics.reporter.prom.class:
org.apache.flink.metrics.prometheus.PrometheusReporter
    kubernetes.operator.metrics.reporter.prom.port: 9999

metrics:
  port: 9999
```

Deploy the operator with:

```
Shell
   helm upgrade --install cp-flink-kubernetes-operator <path_to_chart>/
   -n flink -f <path_to_values.yaml>
```

2. Create a PodMonitor for Operator Metrics

```
Unset
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: flink-kubernetes-operator-pod-monitor
  labels:
    release: prometheus
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: flink-kubernetes-operator
  namespaceSelector:
    matchNames:
      - flink-operator
  podMetricsEndpoints:
    - port: metrics
      path: /metrics
      interval: 30s
```

Apply with:

```
Shell
   kubectl apply -f flink-kubernetes-operator-pod-monitor.yaml
```

Prometheus Metrics Configuration for Flink Applications

1. Configure Prometheus Reporter in Flink Application

Edit the flink-conf.yaml used by your Flink jobs:

```
Unset
spec:
 flinkConfiguration:
   metrics.reporter.prom.factory.class:
"org.apache.flink.metrics.prometheus.PrometheusReporterFactory"
   metrics.reporter.prom.port: "9249"
   rest.profiling.enabled: "true"
```

Expose Metrics Port via PodTemplate in FlinkDeployment

```
Unset
spec:
  podTemplate:
    spec:
      containers:
        - name: flink-main-container
          ports:
            - containerPort: 9249
              name: prometheus
```

2. Create PodMonitors for Flink JobManager and TaskManager

```
Unset
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: flink-application-pod-monitor
  labels:
    release: prometheus
spec:
  selector:
    matchLabels:
      app: flink-app
  namespaceSelector:
    matchNames:
      - flink
  podMetricsEndpoints:
    - port: prometheus
      path: /metrics
      interval: 30s
```

Apply with:

```
Shell
kubectl apply -f flink-application-pod-monitor.yaml
```

Visualizing Metrics with Grafana

Grafana is deployed with Prometheus via the kube-prometheus-stack Helm chart.

Access Grafana:

```
Shell
kubectl port-forward deployment/prometheus-grafana 3000
```

Go to http://localhost:3000, login (default: admin/admin).

Import Dashboards

Please refer here for grafana dashboard:
https://github.com/Irtebat/flink-k8s-deployment/tree/master/misc-flink_monitoring_setup/flink-dashboard.json

Flink Monitoring REST API Integration

Flink metrics can be queried via the Monitoring REST API.

```shell
kubectl port-forward svc/flink-app-rest 8081:8081 -n flink
```

Note: This may be exposed as a nodeport

Entity-Specific Metrics

- /jobmanager/metrics
- /taskmanagers/<taskmanagerid>/metrics
- /jobs/<jobid>/metrics
- /jobs/<jobid>/vertices/<vertexid>/subtasks/<subtaskindex>

Aggregated Metrics (All Entities)

- /taskmanagers/metrics
- /jobs/metrics
- /jobs/<jobid>/vertices/<vertexid>/subtasks/metrics
- /jobs/<jobid>/vertices/<vertexid>/jm-operator-metrics

Aggregated Metrics (Filtered Subsets)

- /taskmanagers/metrics?taskmanagers=A,B,C
- /jobs/metrics?jobs=D,E,F
- /jobs/<jobid>/vertices/<vertexid>/subtasks/metrics?subtask=1,2,3

Read more here : https://nightlies.apache.org/flink/flink-docs-release-2.0/docs/ops/rest_api/