Broad Functionality: **Resource consumption monitoring**

Principle: **Governance**

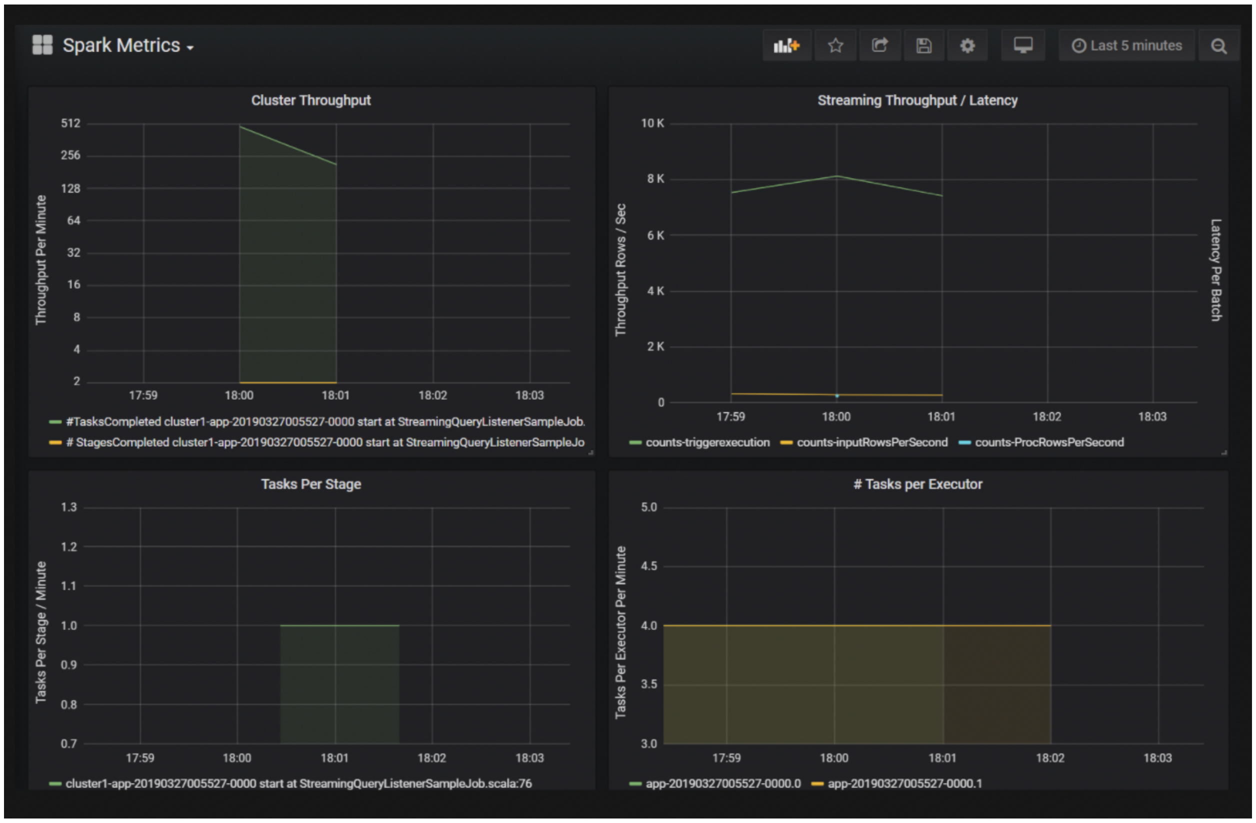
Understanding Functionality: MLflow should provide a view of platform usage across the organization, including which users and models are consuming computing power, enabling effective resource planning. The Resource Consumption Monitor should also provide visibility of active modeling workers across the clusters, providing general information about the current state of the application and specific information about the status of components. With this service in place, you should be able to track the usage of each project. Specifically, the Resource Monitor provides the number of currently running jobs, number of allowed concurrent jobs, and number of jobs waiting for a worker. Additionally, the tool provides information on which specific users are employing system resources.

Does MLflow supports the functionality *directly*: **No**

Does MLflow supports the functionality *indirectly*: **Yes**

- Explanation: There is no way we can do it directly on the local machine using MLflow. However, Databricks and Azure provide the frameworks that are Azure Monitor and Grafana, which can be integrated with our running clusters. Both the Azure Log Analytics and Grafana dashboards include a set of time-series visualizations. Each graph is a time-series plot of metric data related to an Apache Spark job, stages of the job, and tasks that make up each stage. Below are the few metrics that can be monitored:
  A. Job Latency
  B. Stage Latency
  C. Sum task execution per host
  D. Task metric
  E. Cluster throughput
  F. Streaming latency
  G. Resource consumption per executor
- Extra remark: Though a lot of metrics can be monitored using above mentioned integration but most of the features that we are looking for from resource consumption monitoring like, consumption at the user or project level may not be available.

Screenshots for MLflow supports the functionality *directly* or *indirectly*:


Spark Metrics
Last 5 minutes
Cluster Throughput
Throughput Per Minute
512
256
128
64
32
16
8
4
2
17:59
18:00
18:01
18:02
18:03
#TasksCompleted cluster1-app-20190327005527-0000 start at StreamingQueryListenerSampleJob.
# StagesCompleted cluster1-app-20190327005527-0000 start at StreamingQueryListenerSampleJo
Streaming Throughput / Latency
Throughput Rows / Sec
Latency Per Batch
10 K
8 K
6 K
4 K
2 K
0
counts-triggerexecution
counts-inputRowsPerSecond
counts-ProcRowsPerSecond
Tasks Per Stage
Tasks Per Stage / Minute
1.3
1.2
1.1
1.0
0.9
0.8
0.7
cluster1-app-20190327005527-0000 start at StreamingQueryListenerSampleJob.scala:76
# Tasks per Executor
Tasks Per Executor Per Minute
5.0
4.5
4.0
3.5
3.0
app-20190327005527-0000.0
app-20190327005527-0000.1