Cloudera DataFlow

Monitoring Flow Deployments

Date published: 2021-04-06
Date modified: 2024-01-09

https://docs.cloudera.com/

Legal Notice

Contents

Introduction to monitoring flow deployments

The Cloudera DataFlow UI is the central monitoring console for all your deployments across environments. It is the space where you can monitor flow metrics and infrastructure usage, and manage deployments. You can also use reporting tasks to monitor flow deployments using external tools.

In the Cloudera DataFlow UI, you can monitor the status of your deployment and whether the deployment meets performance expectations. You can filter and sort your deployments based on your requirements. You can also fetch details for your deployments for a specified time period and refresh deployment data.

You can monitor the key performance indicators (KPIs), system metrics, and alerts set for your deployments in the Cloudera DataFlow UI. You get alerts for deployments breaching certain assigned KPIs and get recommendations about how to tune your deployments. You also receive alerts if there are issues in your deployment.

After you deploy a flow definition, you can monitor the deployment in the Dashboard of Cloudera DataFlow.

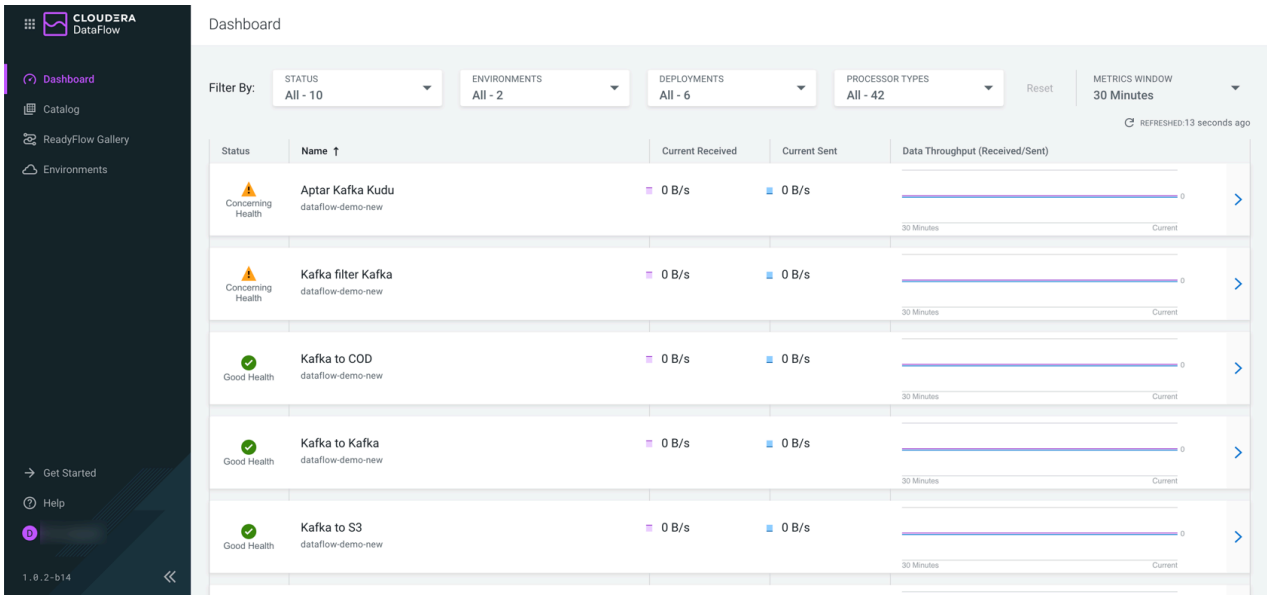
You can also create reporting tasks to monitor health and performance of your flow deployments using external systems.

Monitor health of deployments

You can monitor the health of your deployments in the Dashboard page of the Cloudera DataFlow UI. By monitoring health of deployments, you can track the status of the deployment, the amount of data received and sent by the deployment.

You can perform the following tasks in the Dashboard page:

- Monitor health of deployments
- Filter and sort deployments
- Fetch metrics details for deployments
- Refresh deployments data



Monitor health of deployments

Monitor the status, amount of data sent and received details for your deployments. You can also view the details of the amount of data sent and received in a graphical format.

You can monitor the following details of your deployments in the Dashboard page:

- Status

  Shows status of your deployments including Good Health, Terminating, Updating, Unknown, Deploying, Suspended, Concerning Health, and Bad Health. You can also click Status to sort the status of your deployments as required.
- Name

  Shows name of your deployments. You can also click Name to sort the names of your deployments alphabetically if required.
- Data Received

  Shows the amount of data that your deployments receive from the source. You can also click Data Received to sort your deployments in increasing or decreasing amounts of data received by the deployments.
- Data Sent

  Shows the amount of data that your deployments send to the targets. You can also click Data Sent to sort your deployments in increasing or decreasing amounts of data sent by the deployments.
- Received/Sent Stream Graph

  Shows graphical representation of data received and sent by your deployments.

Filter and sort deployments

Filter the deployments according to your requirement.

You can filter your deployments by:

- Status

  Status of deployments can be of many types including Bad Health, Concerning Health, Good Health, Deploying, Starting, Suspending, Suspended, Terminating, Unknown, and Updating. By default, all statuses are selected.
- Environments

  You can either select all environments or a specific environment which is enabled. By default, all environments are selected.
- Deployments

  You can use this filter to search your flow deployments by name. You can filter your deployments by either selecting all deployments or a specific deployment. By default, all deployments are selected.
- Processor Types

  You can filter your deployments by either selecting all processor types or a specific processor. Processors can be of many types including Apache, Avro, CSV, JSON, Kafka, and SQL. By default, all processor types are selected.

Click Reset to set the filter options to their default values.

You can sort deployments by:

- Status
- Name
- Data Received
- Data Sent

Click on the column name to sort by ascending or descending order.

Use the Items per page option, at the bottom of the screen, to display the desired number of deployments in a page.

Fetch metrics details for deployments

The Metrics Window option enables you to fetch details for your deployments for a specified time. For example, if you select 30 minutes in the Metrics Window, the Cloudera DataFlow UI displays details of your deployments for the last 30 minutes, and if you select 1 hour, the UI displays details of your deployments for the last one hour. You can also select 12 or 24 hours based on your requirement.

Refresh deployments data

By default, every 30 seconds all data for your deployments are updated. Click the circle-arrow icon to refresh the data.

Monitor performance of deployments

You can monitor performance of your deployments in the Dashboard page of the Cloudera DataFlow UI.

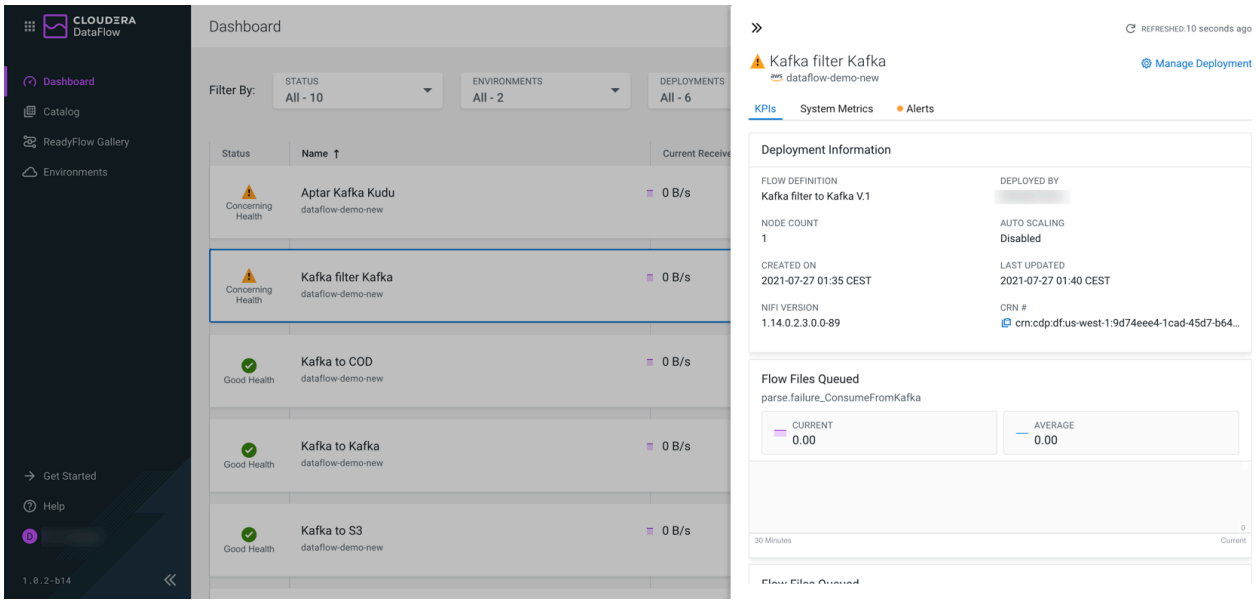
To monitor performance, you can monitor the KPIs, system metrics, and alerts set for your deployments in the Cloudera DataFlow UI. Click on each deployment to get the details of KPIs, system metrics, and alerts. Alternately, you can click on the arrow placed at the end of each deployment row.

Key performance indicators for NiFi flow deployments

Apache NiFi has multiple metrics to monitor the system such as memory usage, CPU usage, data flow statistics, and so on. Key Performance Indicators (KPIs) are representations of those metrics for a NiFi component in Cloudera DataFlow. You can use KPIs to monitor critical parts of your NiFi deployments on the central monitoring dashboard of Cloudera DataFlow.

For more information on KPIs, see *KPI Overview* and *Using KPIs*.

Click the deployment to view the deployment details.



On the KPIs tab of the Deployment Details pane, you can view the deployment information and monitor the KPIs that are defined for your deployment.

You can view the following details in Deployment Information section:

- Flow Definition
- Node Count

- Auto Scaling
- Last Updated
- NiFi Version
- CRN

Depending on the KPIs defined, you can track the following values of the metrics in the deployment:

- Current
- Average
- Boundary

Related Information

Introduction to KPIs

Working with KPIs

System metrics for NiFi flow deployments

Cloudera DataFlow provides comprehensive NiFi cluster-specific metrics on core allocation, disk capacity utilisation, CPU utilisation, and disk usage. You can investigate these system metrics to identify and resolve performance issues related to your NiFi deployments.

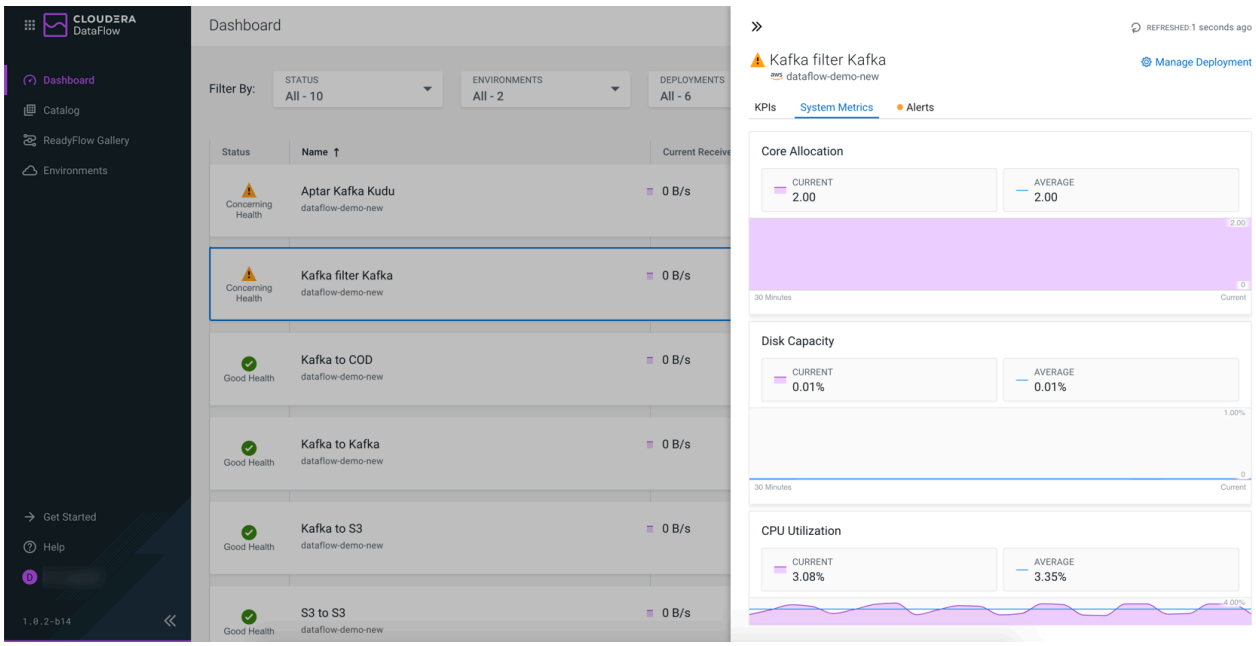
Everytime you deploy and run a NiFi flow, you create a NiFi cluster. The system metrics are details about the NiFi cluster that provide an opportunity to understand the flow deployment status.

These metrics help you to understand the performance of a flow. For example, a flow running at 50% CPU utilization may be an expected (and fine) behavior compared to another flow running at 5% CPU utilization. You are the best judge for how your flow should perform and what thresholds for these metrics dictate that. One metric that you specifically monitor is CPU Utilization. If the value goes above the threshold of 80%, you must scale the cluster.

You can set the key performance indicators (KPIs), with knowledge of your flows, which allow alerts to be sent if certain metrics meet a particular threshold. You might detect situations that are not errors but notes on the state of the system that you need to be aware of. For example, a cluster size reaching the maximum value that is allowed.

The values you measure for metrics depend on the hardware chosen during the flow deployment. For example, the NiFi node size you select (Extra Small to Large) during the flow deployment. These selections enable a deployment to access CPU, memory, or storage which can cause numbers measured to change.

In the Cloudera DataFlow web interface, go to Dashboard and click a deployment to view core allocation, disk capacity, CPU utilization, and disk usage metrics on the System Metrics tab of the Deployment Details pane.



- Core Allocation

  Description: The number of cores that are currently allocated to NiFi containers for the deployment.

  Unit: vCores

  Functionality: Monitors current and average core allocation. You can move your mouse on the graph and get core allocation details for a specific point in time.

  Importance: Helps to assess if the number of cores a flow is using is within the expected range.

  Example: Together with the size selected when the NiFi flow is deployed, the metric reflects the real size of the cluster in terms of allocated CPU resources. For example, when Extra Small size is selected, each NiFi node is capped at 2 vCores. Therefore, a value of 6 means 3 NiFi nodes are currently running to execute the flow. The value fluctuates when the NiFi cluster is auto-scaled up or down.
- Disk Capacity

  Description: Percentage of total allocated storage capacity that is being used by the deployment.

  Unit: Percentage

  Functionality: Monitors current and average disk capacity. You can move your mouse on the graph and get disk capacity details of a specific point in time.

  Importance: Helps to identify if a disk is being under utilized or over utilized; however, again it depends on the expected behavior of a flow.

  Example: This value reflects the storage pressure of the NiFi flow, including Content, Provence, and Flow File repositories. A high percentage can indicate that the traffic is being set in the NiFi cluster, for example, due to failure in delivering the output to the destination system.

- CPU Utilization

  Description: The ratio of the total CPU time consumed by the deployment in the past 5 minutes over the total allocated CPU quota for the deployment.

  Unit: Percentage

  Functionality: Monitors current and average CPU utilization. You can move your mouse on the graph and get the CPU utilization details for a specific point in time.

  Importance: Helps to identify if a flow is using more or less CPU than expected.

  Example: The value indicates the computation pressure of the NiFi cluster running the deployed flow. In a typical case, when auto-scaling is enabled and the CPU utilization reaches the threshold internally defined by Cloudera DataFlow, more NiFi instances are brought up and join the cluster to share the workload, before the value starts dropping.
- Disk Usage

  Description: The total amount of bytes saved on the storage volumes of the deployment.

  Unit: GB, MB, and so on depending on the scale of the value

  Functionality: Monitors current and average disk usage. You can move your mouse on the graph and get the disk usage details for a specific point in time.

  Importance: Provides visibility of how much disk is being used by the deployment or Kubernetes node. It can be used to assess what is utilizing a lot of space.

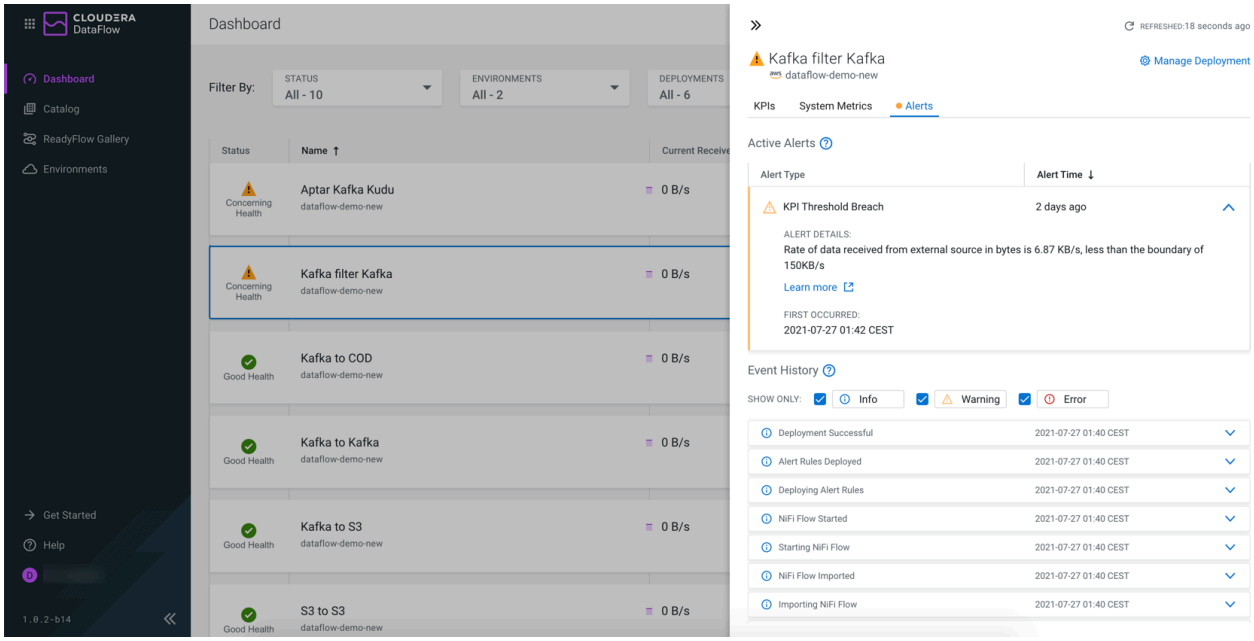
  Example: Enables you to know the current absolute amount of on-disk data of the whole deployment, which may indicate how long an offloading can take.

Alerts for NiFi flow deployments

You can set alerts while configuring KPIs in your deployment. If you set an alert for a KPI and metric combination, you get alerted if your flow definition meets the conditions set in that alert.

Click the deployment to view the deployment details.

On the Alerts tab of the Deployment Details pane, you can monitor the active alerts and event history details.



In the Active Alerts section, you can track the alert types and time of the alert. If you click on the alert you want to track, you can see the alert details and the exact time when the alert occurred. Active alerts notify you of a current condition that may require attention. If you set an alert for a deployment-related KPI, you get alerted here as well. Active alerts automatically become inactive and move to the Event History section if the condition is not met anymore.

Tip:

KPI Alerts are suspended when a flow deployment is suspended. They are reactivated when the flow deployment is restarted. Suspending a flow deployment also resolves any Active KPI alerts.

In the Event History section, you can track event-related information and past alert conditions such as when the deployment was initiated and whether the deployment was successful. You can filter the event history list to only show events of a specific type including information, warning, and error.

Monitor auto-scaling events

Every auto-scaling operation is logged as a flow deployment event. These events can be monitored through either the UI or CDP CLI.

For UI

In the **Dashboard**, click on the flow deployment that you want to monitor and perform the following steps:

1. In the side panel displaying Deployment Information, click on the Alerts tab.
2. Look for Deployment Pod Count Changed events.
3. Filter for informational events by only selecting the Info category.
4. Click Load More to see older events.

For CLI

To monitor auto-scaling events via the CDP CLI, execute the `cdp df list-deployment-events` command with the following configuration:

```
cdp df list-deployment-events --deployment-crn [***YOUR DEPLOYMENT CRN***]
```

Replace *[***YOUR DEPLOYMENT CRN***]* with the actual CRN of the deployment for which you want to monitor auto-scale events.

The command returns all deployment events in a JSON array that you can parse for Deployment Pod Count Cha nged events:

```
{
    "eventSummaries": [
        {
            "crn": "crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d512
59233:event:DEPLOYMENT/36761c1d-264b-46ee-8a20-34ccedf2bc90/042cc645-0d8
e-4d1e-aebd-2c90e665f845",
            "name": "Deployment Pod Count Changed",
            "severity": "INFO",
            "firstOccurrence": 1687314605747,
            "eventType": "STATUS_UPDATE"
        },
        {
            "crn": "crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d5125
9233:event:DEPLOYMENT/36761c1d-264b-46ee-8a20-34ccedf2bc90/20c9361f-d33e
-481a-98c2-0b62e3666798",
            "name": "Deployment Successful",
            "severity": "INFO",
            "firstOccurrence": 1687314062456,
```

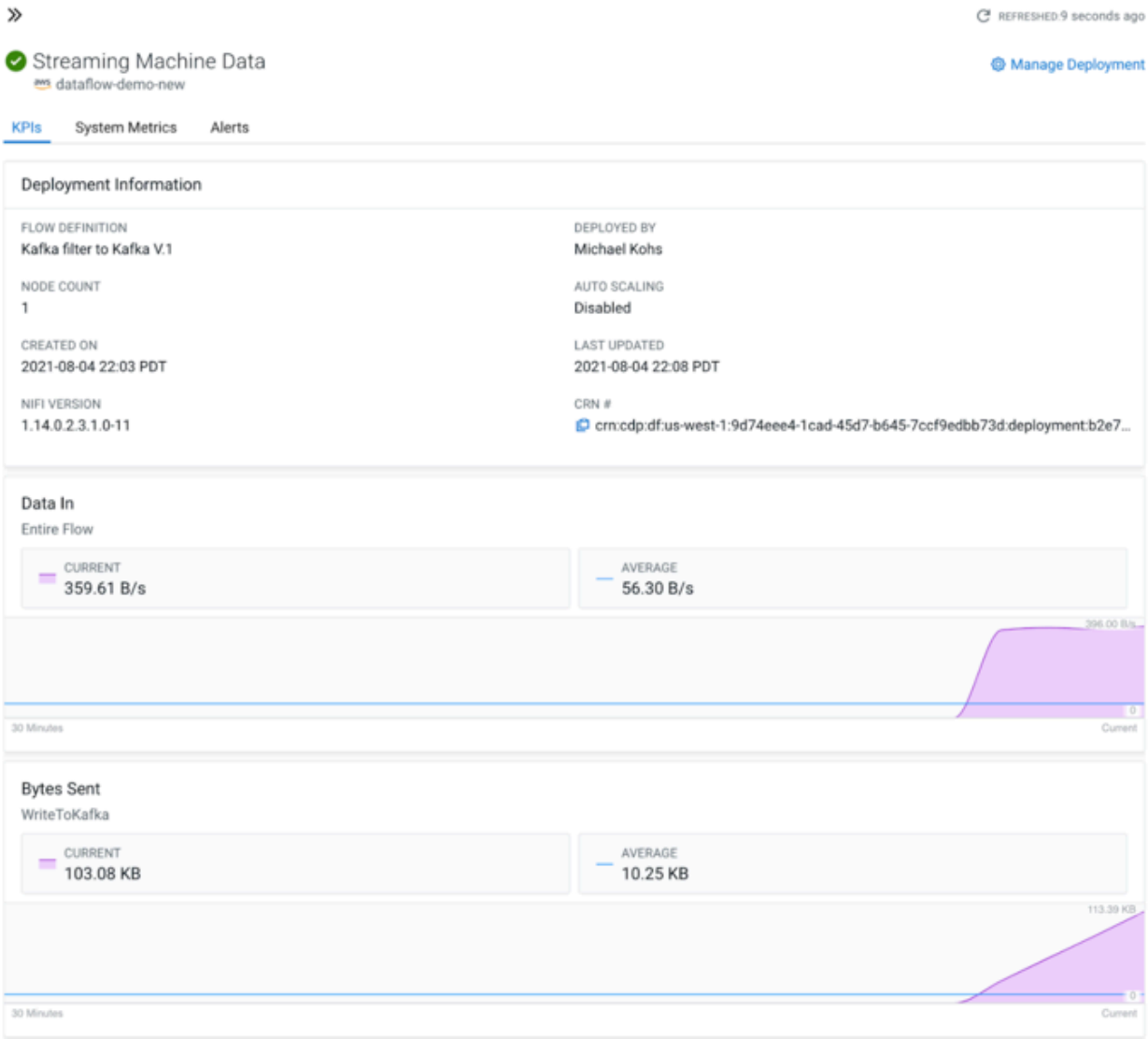
```
            "eventType": "STATUS_UPDATE"
        }
    ]
}
```

Key performance indicators

You can use key performance indicators (KPIs) to monitor critical parts of your NiFi deployments on the central monitoring dashboard. You do not need to drill deep into the NiFi data flow and find the metric to monitor it in NiFi. You can also choose to create alerts for your KPIs in Cloudera DataFlow.

Apache NiFi has multiple metrics to monitor for the system such as memory usage, CPU usage, data flow statistics and so on. KPIs are representations of those metrics for a NiFi component in Cloudera DataFlow.

You can set alerts while configuring KPIs in your deployment. If you set an alert for a KPI scope and metric combination, you get alerted if your flow definition meets the conditions set in that alert.



In the above image, a KPI is set for the Entire Flow KPI scope and Data In metric combination. When the data into the entire flow exceeds the 1MB boundary, an alert is created. No alert boundary is set for the Processor KPI scope and Average Linear Duration metric combination.

Working with KPIs

Learn multiple scenarios where you can view how to use KPIs to monitor critical parts of your NiFi deployments on the central monitoring dashboard.

Use KPIs to monitor whether your data flow is processing data

Learn how to use KPIs to monitor whether your data flow is processing data.

Scenario

You want to track how much data your flow is receiving from external systems or processing to an external system. It is a common use case where you might want to track whether your flow is reading or writing data as expected.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select Entire Flow KPI scope and Data In or Data Out metric to track. Select Data In metric to track how much data your flow is reading and Data Out metric to track how much data your flow is writing to the external system. In this example, Data In metric is selected. The Data In metric tracks rate of data received from an external source in bytes.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

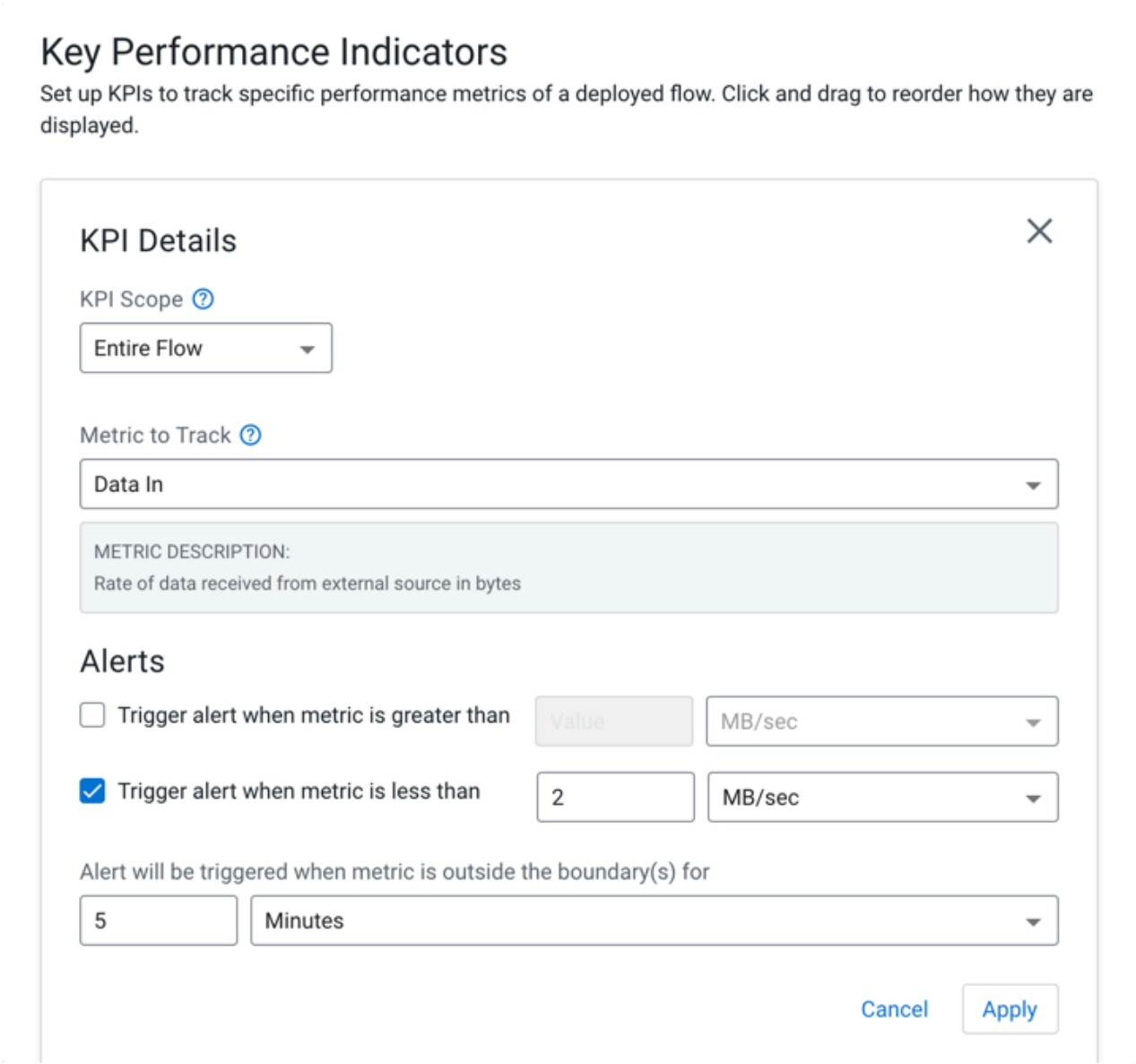
Procedure

1. Select Entire Flow as the KPI scope.
2. Select Data In as the metric to track.
3. Configure the alert settings as per your requirement or when you want to be alerted.
4. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, you get an alert when your flow is receiving less than 2 MB data per second for 5 minutes.



Use KPIs to track the processing latency of streaming data

Learn how to use KPI to track the processing latency of streaming data.

Scenario

Your flow starts reading data from an external system and ends writing the data to some other place. In between all processing is done. A common question or a common performance indicator that you want to track might be how long does it take for the data to be processed.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select Processor KPI scope and Average Lineage Duration metric to track. The Average Lineage Duration metric tracks the time elapsed between when the original source data was received and the time when the processing event occurred.

Note: Average Lineage Duration does not always measure processing latency of a flowfile from the very first processor to the last processor.

If you have a simple NiFi flow, where none of the processors creates a new flowfile in between, Average Lineage Duration measures the processing latency of the flowfile from the first to the last processor.

But, if a processor in your NiFi flow creates a new flowfile in between, Average Lineage Duration measures the processing latency from that very processor, where a new flow file is created, to the last processor.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

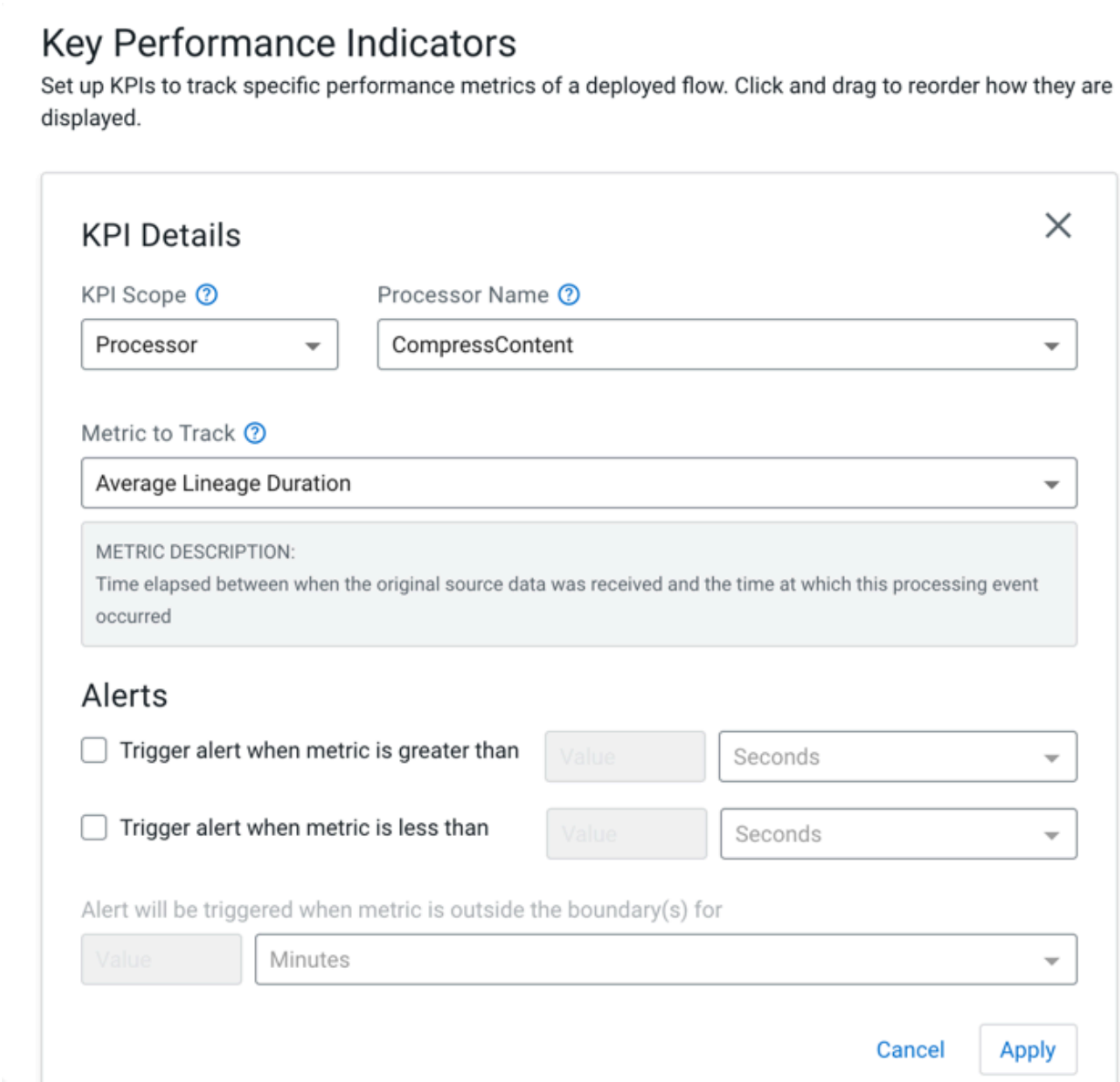
Procedure

1. Select Processor as the KPI scope.
2. Select the processor, you want to monitor, in the Processor Name field.
3. Select Average Lineage Duration as the metric to track.
4. Configure the alert settings as per your requirement or when you want to be alerted.
5. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, Processor KPI scope and Average Lineage Duration metric to track are selected. No alert is set here.



Use KPIs to monitor connection usage

Learn how to use KPIs to monitor connection usage.

Scenario

You want to track errors in a failure queue. You read data from a Kafka topic and apply a schema to it. The schema does not fit the message and the operation fails. You do not want to process the message and route it to a failure queue. Now, you want to track data quality in your deployment.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. You need to create a KPI on this queue to track failure. Select Connection KPI scope, connection name, and Percent Full metric to track. The Percent Full metric indicates how much of the queue's capacity is used for your data flow.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

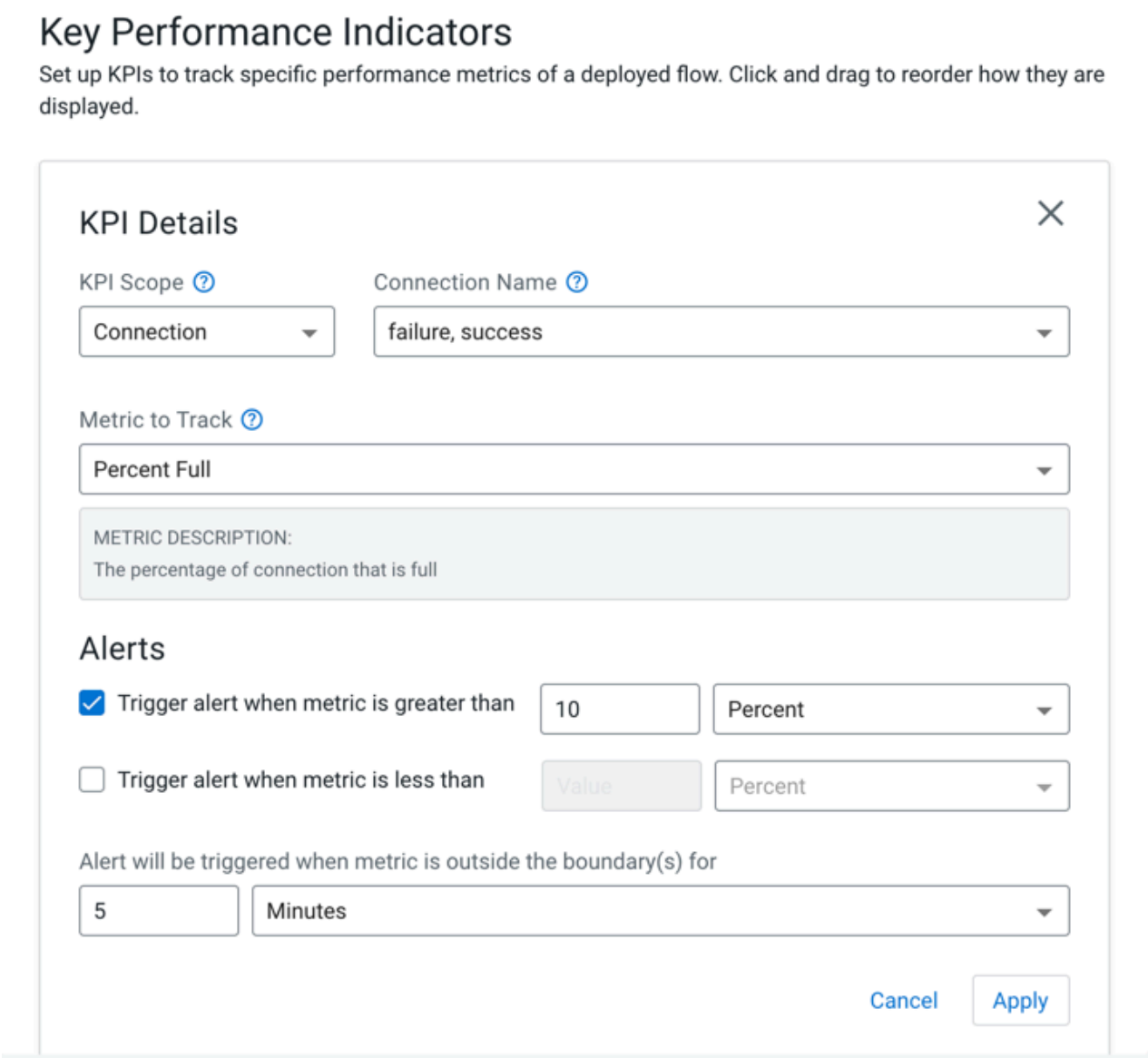
Procedure

1. Select Connection as the KPI scope.
2. Select Percent Full as the metric to track.
3. Select the failure queue, you created, in the Connection Name field.
4. Configure the alert settings as per your requirement or when you want to be alerted.
5. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, you get an alert when 10 percent of the queue's capacity is used for storing failure messages for 5 minutes.



Use KPIs to monitor individual processor metrics

Learn how to use KPIs to monitor individual processor metrics.

Scenario

You want to track a specific metric for a processor in your data flow.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select Processor KPI scope, the processor you want to monitor in the Processor Name field, and the metric you want to track. In this example, Bytes Sent metric is selected. The Bytes Sent metric tracks number of bytes sent to an external recipient.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

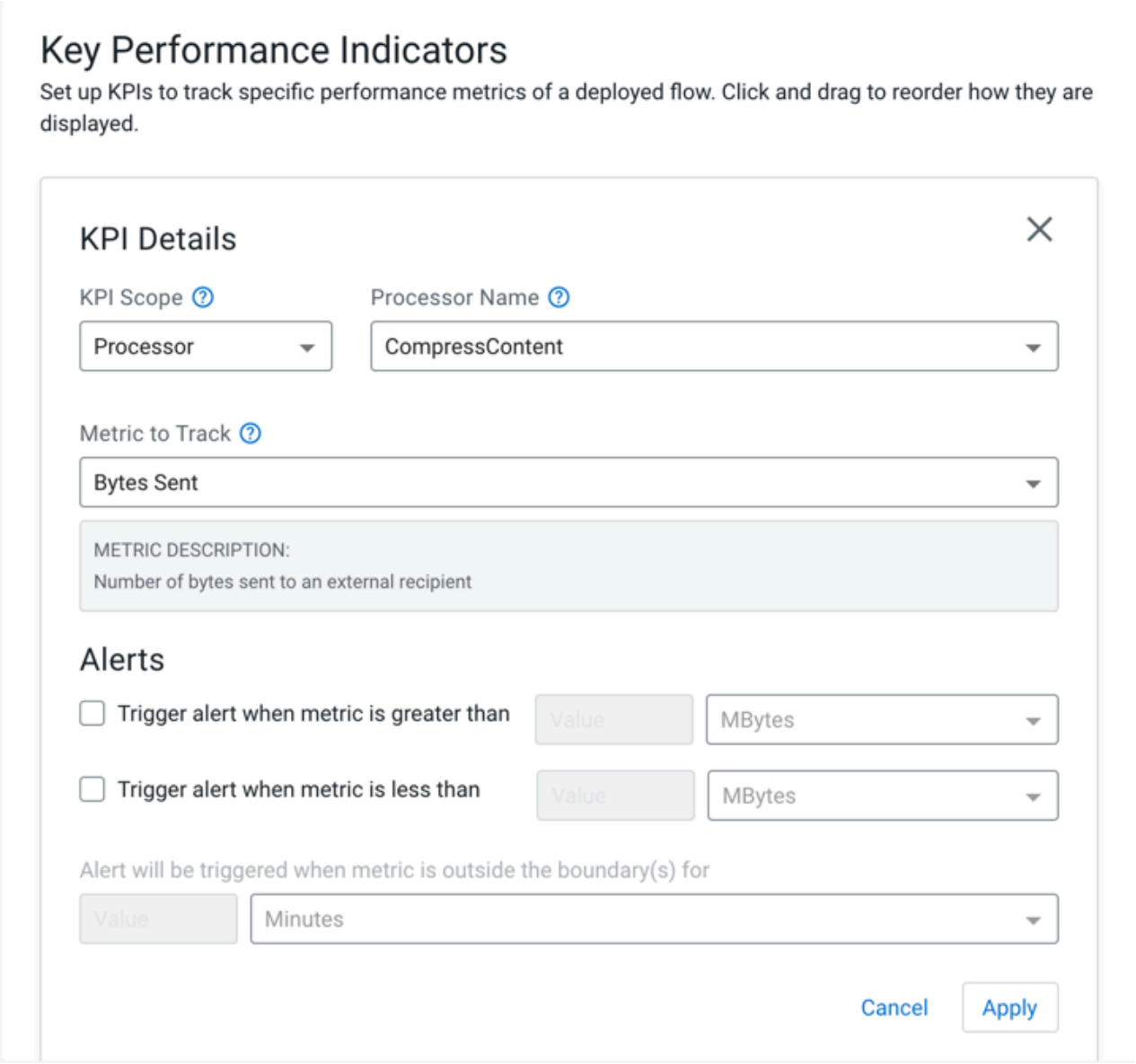
Procedure

1. Select Processor as the KPI scope.
2. Select Bytes Sent as the metric to track.
3. Configure the alert settings as per your requirement or when you want to be alerted.
4. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, Processor KPI scope and Bytes Sent metric to track are selected. No alert is set here.



Use KPIs to monitor auto-scaling operations

Learn how to use KPIs to monitor auto-scaling operations.

Scenario

You want to use KPIs to track auto-scaling operations.

Solution

You can achieve your goal by selecting appropriate KPI scope and metric while deploying your flow in Cloudera DataFlow. Select System KPI scope and Core Allocation metric to track. Core Allocation metric tracks the numbers of cores allocated. After the flow deployment, you can view a graph that represents how many cores are used right now and you can also view when it scales up or scales down.

Before you begin

You must have reached the Add New KPI screen while deploying your flow in Cloudera DataFlow.

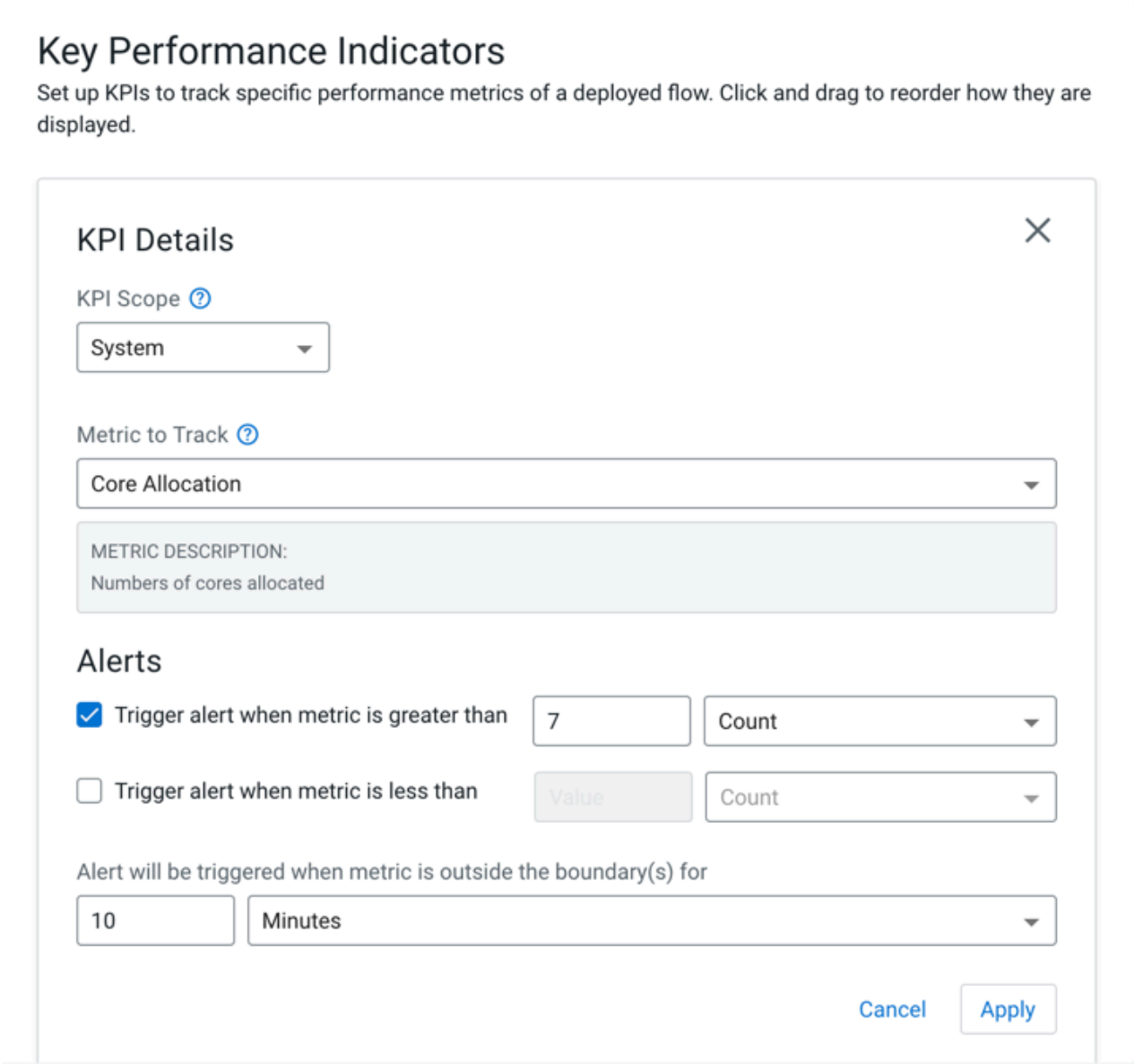
Procedure

1. Select System as the KPI scope.
2. Select Core Allocation as the metric to track.
3. Configure the alert settings as per your requirement or when you want to be alerted.
4. Click Apply.

   Similarly, you can set other metrics and alerts as per your requirement.

Example

In this example, you get an alert when the number of cores allocated for your flow is greater than seven.



KPI scope types in Cloudera DataFlow

KPI scope specifies the focus area of the KPI for the flow deployment. You can select individual NiFi components such as a processor or connection, or larger components such as a process group or the entire flow.

You can track the different statistics of a NiFi data flow by using the following KPI scopes:

- Connection
- Processor
- Process Group
- Entire Flow
- System

Connection KPIs

Connection KPIs expose the metrics that NiFi provides for the connection used in the NiFi data flow. Choosing Connection allows you to set alerts based on any of the connection metrics that the data flow monitors.

Metric name: Bytes Queued

Metric unit: Bytes

Meaning: How much data is stored in a queue?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data queued in the selected queue or connection is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Percent Full

Metric unit: Percentage

Meaning: How much of the queue's capacity is used?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data queued in the selected queue or connection is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Flow Files Queued

Metric unit: Count

Meaning: How many flow files are queued in this connection?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles queued for the selected queue or connection is either higher than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Processor KPIs

Processor KPIs expose the metrics that NiFi provides for the processors. Choosing Processor allows you to set alerts based on any of the processor metrics that the data flow monitors.

Metric name: Average Lineage Duration

Metric unit: Seconds

Meaning: How much time elapsed between when the original source data was received and the time at which this processing event occurred?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the average lineage duration for all flowfiles within the specified time period is either above the upper boundary or below the lower boundary.

Metric name: Bytes In

Metric unit: Bytes

Meaning: How much data is received by the selected processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data received by the selected processor is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Bytes Out

Metric unit: Bytes

Meaning: How much data is sent by the selected processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data sent by the selected processor is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Bytes Queued

Metric unit: Bytes

Meaning: How much content is queued for the selected processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data queued for the selected precessor is either more than what is specified in the upper boundary or less than specified in the lower boundary for the given time period.

Metric name: Bytes Received

Metric unit: Bytes

Meaning: How much data is received from a source that is external to the flows?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data received or read by the selected precessor is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Bytes Sent

Metric unit: Bytes

Meaning: How much data is sent to an external recipient?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the data sent by the selected precessor is either more than what is specified in the upper boundary or less than specified in the lower boundary for the given time period.

Metric name: Flowfiles Input to Processor

Metric unit: Count

Meaning: How many flowfiles are received from a source within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the number of flowfiles received from a source within the flow is either more than the specified in the upper boundary or less than the specified in the lower boundary for the given time period.

Metric name: Flowfiles Out from Processor

Metric unit: Count

Meaning: How many flowfiles are sent out of the processor?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the number of flowfiles sent out of the processor is either more than the specified in the upper boundary or less than the specified in the lower boundary for the given time period.

Metric name: Flowfiles Received

Metric unit: Count

Meaning: How many flowfiles are received by the selected processor from a source that is external to the flows?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles received by the selected processor is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Flowfiles Sent

Metric unit: Count

Meaning: How many flowfiles are sent by the selected processor to an external recipient?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles sent by the selected processor is either below the specified lower threshold or above the upper threshold for the given time period.

Process Group KPIs

Process Group KPIs expose the metrics that NiFi provides for the process groups. Choosing Process Group enables you to set alerts based on any of the process group metrics that the data flow monitors.

Metric name: Bytes Queued

Metric unit: Bytes

Meaning: How much data is queued, in total, in queues of the selected process group?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data queued for the selected process group is either more than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Bytes Received

Metric unit: Bytes

Meaning: How much data is received by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data received by the selected process group is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Bytes Sent

Metric unit: Bytes

Meaning: How much data is sent by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data sent by the selected process group is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Flowfiles Input

Metric unit: Count

Meaning: How many flowfiles are received by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles received by the selected process group is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Flowfiles Out

Metric unit: Count

Meaning: How many flowfiles are sent by the selected process group within the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles sent by the selected process group is either below the specified lower threshold or above the specified upper threshold for the given time period.

Metric name: Flowfiles Queued

Metric unit: Count

Meaning: How many flowfiles are queued, in total, in queues for the selected process group?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles queued for the selected process group is either higher than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

Metric name: Flowfiles Received

Metric unit: Count

Meaning: How many flowfiles are received from a source that is external to the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles received from a source that is external to the flow is either above the specified in the upper boundary or below the specified in the lower boundary.

Metric name: Flowfiles Sent

Metric unit: Count

Meaning: How many flowfiles are sent to an external recipient?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles sent to an external recipient is either above the specified in the upper boundary or below the specified in the lower boundary.

Entire Flow KPIs

Entire Flow KPIs expose the metrics that NiFi provides for the entire data flow. Choosing Entire Flow enables you to set alerts based on any of the flow metrics that the data flow monitors.

Metric name: Data In

Metric unit: Bytes/sec

Meaning: How much data did the entire flow receive from external systems per second?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data received for the entire flow is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Data Out

Metric unit: Bytes/sec

Meaning: How much data did the entire flow send to external systems per second?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated data sent for the entire flow is either below the specified lower threshold or above the upper threshold for the given time period.

Metric name: Flow Files Queued

Metric unit: Count

Meaning: How many flow files are queued in all connections of the flow?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the aggregated number of flowfiles queued for all connections of the flow is either higher than what is specified in the upper boundary or less than what is specified in the lower boundary for the given time period.

System KPIs

System KPIs expose the metrics that NiFi provides for the system. Choosing System allows you to set alerts based on any of the system metrics that the data flow monitors.

Metric name: CPU Utilization

Metric unit: Percentage

Meaning: How much of the available CPU to the entire cluster is being used by deployment?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the CPU utilization is either above the specified upper threshold or below the specified lower threshold for the given time period. For example, if the threshold is 80% and the time period is 5 minutes, all data points that are being captured within five minutes must show a higher utilization than 80%, to trigger an alert.

Metric name: Core Allocation

Metric unit: Count

Meaning: How many cores are allocated?

Threshold: Upper and lower boundary

You can use this KPI to create alerts when the core allocation is either above the specified upper threshold or below the specified lower threshold for the given time period.

Reporting tasks

Reporting tasks allow you to monitor flow deployments using external applications.

Before you begin

- If you want to create a reporting task using a custom NAR, you have added a custom NAR to the flow during deployment.
- You have DFFlowAdmin role for the environment where you want to create the reporting task.

Create a JSON file

To create a reporting task, use the following template:

Replace elements marked *[***SOME VALUE***]* with actual values valid in your environment. You can specify additional properties that may be necessary to your specific use case.

```
{
  "name": "[***REPORTING TASK NAME***]",
  "type": "[***FULLY QUALIFIED CLASS NAME***]",
  "properties": {
    "[***PROPERTY NAME 1***]": "[***PROPERTY VALUE 1***]",
    "[***PROPERTY NAME 2***]": "[***PROPERTY VALUE 2***]"
  },
  "scheduledState": "[***RUNNING|DISABLED***]",
  "schedulingPeriod": "[***SCHEDULING PERIOD***]",
  "schedulingStrategy": "[***TIMER_DRIVEN|CRON_DRIVEN***]"
}
```

To define a schedulingPeriod that corresponds to the TIMER_DRIVEN schedulingStrategy, specify a frequency. For example, "5 min" or "45 sec" or "2 hours".

To define a schedulingPeriod that corresponds to the CRON_DRIVEN schedulingStrategy specify, for example, "* * * * * ?"

By setting scheduledState to RUNNING, the task automatically starts upon creation. Otherwise, it remains stopped.

For example:

```
{
  "name": "My Reporting Task",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "scheduledState": "RUNNING",
  "schedulingPeriod": "5 min",
  "schedulingStrategy": "TIMER_DRIVEN"
}
```

If there is more than one reporting task with the same class name in the deployment, you can select the required one using a "bundle", as the following example shows:

```
{
  "name": "ControllerStatusReportingTask",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.23.2.2.3.12.0-7"
  },
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "scheduledState": "RUNNING",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN"
}
```

Create a reporting task

Assuming you created a file named reporting-task.json with contents similar to one of the above examples, execute the command:

```
dfworkload create-reporting-task --environment-crn [***ENVIRONMENT CRN***]
 --deployment-crn [***DEPLOYMENT CRN***] --file-path reporting-task.json
```

Replace *[***ENVIRONMENT CRN***]* and *[***DEPLOYMENT CRN***]* with actual values valid in your environment.

List all reporting tasks

To list existing reporting tasks, use the following command:

```
dfworkload list-reporting-tasks --environment-crn [***ENVIRONMENT CRN***] --
deployment-crn [***DEPLOYMENT CRN***]
```

Replace *[***ENVIRONMENT CRN***]* and *[***DEPLOYMENT CRN***]* with actual values valid in your environment.

Delete a reporting task

Note:

The command output of dfworkload list-reporting-tasks includes a "crn" field, which you can use to replace *[***REPORTING TASK CRN***]* in the delete command.

To delete a reporting task, use the following command:

```
dfworkload delete-reporting-task --environment-crn [***ENVIRONMENT CRN***]
 --deployment-crn [***DEPLOYMENT CRN***] --reporting-task-crn [***REPORTING
 TASK CRN***]
```

Replace *[***ENVIRONMENT CRN***]*, *[***DEPLOYMENT CRN***]*, and *[***REPORTING TASK CRN***]* with actual values valid in your environment.

Reporting task JSON file reference

These file samples provide a starting point on how to configure reporting tasks for Cloudera DataFlow (CDF). You can use the CDP CLI to add the created reporting tasks to CDF flow deployments.

AmbariReportingTask

```
{
    "name": "AmbariReportingTask",
    "type": "org.apache.nifi.reporting.ambari.AmbariReportingTask",
    "properties": {
      "Application ID": "nifi",
      "Hostname": "${hostname(true)}",
      "Metrics Collector URL": "http://localhost:6188/ws/v1/timeline/metrics
"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
    "name": "AmbariReportingTask",
    "type": "org.apache.nifi.reporting.ambari.AmbariReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-ambari-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Application ID": "nifi",
      "Hostname": "${hostname(true)}",
      "Metrics Collector URL": "http://localhost:6188/ws/v1/timeline/
metrics"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see AmbariReportingTask in the CDF documentation.

AzureLogAnalyticsProvenanceReportingTask

```
{
    "name": "AzureLogAnalyticsProvenanceReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAnalytic
sProvenanceReportingTask",
    "properties": {
      "Job Name": "nifi_reporting_job",
      "Platform": "nifi",
      "Application ID": "nifi",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsights.azur
e.com/api/logs?api-version=2016-04-01",
      "include-null-values": "false",
      "Batch Size": "1000",
      "Log Analytics Custom Log Name": "nifiprovenance",
      "Instance ID": "${hostname(true)}",
      "start-position": "beginning-of-stream",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "AzureLogAnalyticsProvenanceReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAnaly
ticsProvenanceReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-azure-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Job Name": "nifi_reporting_job",
      "Platform": "nifi",
      "Application ID": "nifi",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsight
s.azure.com/api/logs?api-version=2016-04-01",
      "include-null-values": "false",
      "Batch Size": "1000",
      "Log Analytics Custom Log Name": "nifiprovenance",
      "Instance ID": "${hostname(true)}",
      "start-position": "beginning-of-stream",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see AzureLogAnalyticsProvenanceReportingTask in the CDF documentation.

AzureLogAnalyticsReportingTask

```
  {
    "name": "AzureLogAnalyticsReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAnalyti
csReportingTask",
    "properties": {
      "Send JVM Metrics": "false",
      "Job Name": "nifi_reporting_job",
      "Application ID": "nifi",
      "Log Analytics Custom Log Name": "nifimetrics",
      "Instance ID": "${hostname(true)}",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsights.azur
e.com/api/logs?api-version=2016-04-01"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "AzureLogAnalyticsReportingTask",
    "type": "org.apache.nifi.reporting.azure.loganalytics.AzureLogAna
lyticsReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-azure-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Send JVM Metrics": "false",
      "Job Name": "nifi_reporting_job",
      "Application ID": "nifi",
      "Log Analytics Custom Log Name": "nifimetrics",
      "Instance ID": "${hostname(true)}",
      "Log Analytics URL Endpoint Format": "https://{0}.ods.opinsigh
ts.azure.com/api/logs?api-version=2016-04-01"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see AzureLogAnalyticsReportingTask in the CDF documentation.

ControllerStatusReportingTask

```
{
  "name": "ControllerStatusReportingTask",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "ControllerStatusReportingTask",
  "type": "org.apache.nifi.controller.ControllerStatusReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Show Deltas": "true",
    "reporting-granularity": "five-minutes"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see ControllerStatusReportingTask in the CDF documentation.

DataDogReportingTask

```
{
  "name": "DataDogReportingTask",
  "type": "org.apache.nifi.reporting.datadog.DataDogReportingTask",
  "properties": {
    "Metrics prefix": "nifi",
    "Datadog transport": "Datadog HTTP",
    "Environment": "dev"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "DataDogReportingTask",
  "type": "org.apache.nifi.reporting.datadog.DataDogReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-datadog-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Metrics prefix": "nifi",
    "Datadog transport": "Datadog HTTP",
    "Environment": "dev"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see DataDogReportingTask in the CDF documentation.

MetricsEventReportingTask

```
{
  "name": "MetricsEventReportingTask",
  "type": "org.apache.nifi.reporting.sql.MetricsEventReportingTask",
  "properties": {
    "dbf-default-precision": "10",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MetricsEventReportingTask",
  "type": "org.apache.nifi.reporting.sql.MetricsEventReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-sql-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "dbf-default-precision": "10",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MetricsEventReportingTask in the CDF documentation.

MetricsReportingTask

```
{
  "name": "MetricsReportingTask",
  "type": "org.apache.nifi.metrics.reporting.task.MetricsReportingTask",
  "properties": {},
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MetricsReportingTask",
  "type": "org.apache.nifi.metrics.reporting.task.MetricsReportingTas
k",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-metrics-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {},
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MetricsReportingTask in the CDF documentation.

MonitorDiskUsage

```
{
  "name": "MonitorDiskUsage",
  "type": "org.apache.nifi.controller.MonitorDiskUsage",
  "properties": {
    "Directory Display Name": "Un-Named",
    "Threshold": "80%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MonitorDiskUsage",
  "type": "org.apache.nifi.controller.MonitorDiskUsage",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Directory Display Name": "Un-Named",
    "Threshold": "80%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MonitorDiskUsage in the CDF documentation.

MonitorMemory

```
{
  "name": "MonitorMemory",
  "type": "org.apache.nifi.controller.MonitorMemory",
  "properties": {
    "Memory Pool": "G1 Old Gen",
    "Usage Threshold": "65%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "MonitorMemory",
  "type": "org.apache.nifi.controller.MonitorMemory",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-standard-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Memory Pool": "G1 Old Gen",
    "Usage Threshold": "65%"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see MonitorMemory in the CDF documentation.

PrometheusReportingTask

```
{
  "name": "PrometheusReportingTask",
  "type": "org.apache.nifi.reporting.prometheus.PrometheusReportingTask",
  "properties": {
    "prometheus-reporting-task-metrics-endpoint-port": "9092",
    "prometheus-reporting-task-metrics-strategy": "All Components",
    "prometheus-reporting-task-instance-id": "${hostname(true)}",
    "prometheus-reporting-task-client-auth": "No Authentication",
    "prometheus-reporting-task-metrics-send-jvm": "false"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "60 sec",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "PrometheusReportingTask",
    "type": "org.apache.nifi.reporting.prometheus.PrometheusReportin
gTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-prometheus-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "prometheus-reporting-task-metrics-endpoint-port": "9092",
      "prometheus-reporting-task-metrics-strategy": "All Components",
      "prometheus-reporting-task-instance-id": "${hostname(true)}",
      "prometheus-reporting-task-client-auth": "No Authentication",
      "prometheus-reporting-task-metrics-send-jvm": "false"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "60 sec",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see PrometheusReportingTask in the CDF documentation.

QueryNiFiReportingTask

```
{
  "name": "QueryNiFiReportingTask",
  "type": "org.apache.nifi.reporting.sql.QueryNiFiReportingTask",
  "properties": {
    "dbf-default-precision": "10",
    "sql-reporting-include-zero-record-results": "false",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "QueryNiFiReportingTask",
  "type": "org.apache.nifi.reporting.sql.QueryNiFiReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-sql-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "dbf-default-precision": "10",
    "sql-reporting-include-zero-record-results": "false",
    "dbf-default-scale": "0"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see QueryNiFiReportingTask in the CDF documentation.

ReportLineageToAtlas

```
{
  "name": "ReportLineageToAtlas",
  "type": "org.apache.nifi.atlas.reporting.ReportLineageToAtlas",
  "properties": {
    "atlas-authentication-method": "basic",
    "filesystem-paths-level": "FILE",
    "kafka-kerberos-service-name": "kafka",
    "atlas-connect-timeout": "60 sec",
    "atlas-conf-create": "false",
    "provenance-start-position": "beginning-of-stream",
    "atlas-read-timeout": "60 sec",
    "provenance-batch-size": "1000",
    "aws-s3-model-version": "v2",
    "kafka-security-protocol": "PLAINTEXT",
    "nifi-lineage-strategy": "SimplePath"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "ReportLineageToAtlas",
  "type": "org.apache.nifi.atlas.reporting.ReportLineageToAtlas",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-atlas-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "atlas-authentication-method": "basic",
    "filesystem-paths-level": "FILE",
    "kafka-kerberos-service-name": "kafka",
    "atlas-connect-timeout": "60 sec",
    "atlas-conf-create": "false",
    "provenance-start-position": "beginning-of-stream",
    "atlas-read-timeout": "60 sec",
    "provenance-batch-size": "1000",
    "aws-s3-model-version": "v2",
    "kafka-security-protocol": "PLAINTEXT",
    "nifi-lineage-strategy": "SimplePath"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see ReportLineageToAtlas in the CDF documentation.

ScriptedReportingTask

```
{
  "name": "ScriptedReportingTask",
  "type": "org.apache.nifi.reporting.script.ScriptedReportingTask",
  "properties": {
    "Script Engine": "ECMAScript"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "ScriptedReportingTask",
  "type": "org.apache.nifi.reporting.script.ScriptedReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-scripting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "Script Engine": "ECMAScript"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see ScriptedReportingTask in the CDF documentation.

SiteToSiteBulletinReportingTask

```
{
  "name": "SiteToSiteBulletinReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteBulletinReportingTask",
  "properties": {
    "s2s-transport-protocol": "RAW",
    "Platform": "nifi",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "1 min",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "SiteToSiteBulletinReportingTask",
    "type": "org.apache.nifi.reporting.SiteToSiteBulletinReportingTa
sk",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-site-to-site-reporting-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "s2s-transport-protocol": "RAW",
      "Platform": "nifi",
      "include-null-values": "false",
      "Compress Events": "true",
      "Communications Timeout": "30 secs",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "1 min",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see SiteToSiteBulletinReportingTask in the CDF documentation.

SiteToSiteMetricsReportingTask

```
{
  "name": "SiteToSiteMetricsReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteMetricsReportingTask",
  "properties": {
    "s2s-metrics-format": "ambari-format",
    "s2s-transport-protocol": "RAW",
    "s2s-metrics-application-id": "nifi",
    "s2s-metrics-hostname": "${hostname(true)}",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "SiteToSiteMetricsReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteMetricsReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-site-to-site-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "s2s-metrics-format": "ambari-format",
    "s2s-transport-protocol": "RAW",
    "s2s-metrics-application-id": "nifi",
    "s2s-metrics-hostname": "${hostname(true)}",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see SiteToSiteMetricsReportingTask in the CDF documentation.

SiteToSiteProvenanceReportingTask

```
{
  "name": "SiteToSiteProvenanceReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteProvenanceReportingTask",
  "properties": {
    "s2s-transport-protocol": "RAW",
    "Platform": "nifi",
    "include-null-values": "false",
    "Compress Events": "true",
    "Batch Size": "1000",
    "Communications Timeout": "30 secs",
    "start-position": "beginning-of-stream",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
{
  "name": "SiteToSiteMetricsReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteMetricsReportingTask",
  "bundle": {
    "group": "org.apache.nifi",
    "artifact": "nifi-site-to-site-reporting-nar",
    "version": "1.24.0.2.3.12.0-17"
  },
  "properties": {
    "s2s-metrics-format": "ambari-format",
    "s2s-transport-protocol": "RAW",
    "s2s-metrics-application-id": "nifi",
    "s2s-metrics-hostname": "${hostname(true)}",
    "include-null-values": "false",
    "Compress Events": "true",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

For more information, see SiteToSiteProvenanceReportingTask in the CDF documentation.

SiteToSiteStatusReportingTask

```
{
  "name": "SiteToSiteStatusReportingTask",
  "type": "org.apache.nifi.reporting.SiteToSiteStatusReportingTask",
  "properties": {
    "s2s-transport-protocol": "RAW",
    "Platform": "nifi",
    "Component Name Filter Regex": ".*",
    "include-null-values": "false",
    "Compress Events": "true",
    "Batch Size": "1000",
    "Component Type Filter Regex": "(Processor|ProcessGroup|RemoteProcessG
roup|RootProcessGroup|Connection|InputPort|OutputPort)",
    "Communications Timeout": "30 secs",
    "Instance URL": "http://${hostname(true)}:8080/nifi"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "SiteToSiteStatusReportingTask",
    "type": "org.apache.nifi.reporting.SiteToSiteStatusReportingTask",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-site-to-site-reporting-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "s2s-transport-protocol": "RAW",
      "Platform": "nifi",
      "Component Name Filter Regex": ".*",
      "include-null-values": "false",
      "Compress Events": "true",
      "Batch Size": "1000",
      "Component Type Filter Regex": "(Processor|ProcessGroup|Remote
ProcessGroup|RootProcessGroup|Connection|InputPort|OutputPort)",
      "Communications Timeout": "30 secs",
      "Instance URL": "http://${hostname(true)}:8080/nifi"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see SiteToSiteStatusReportingTask in the CDF documentation.

StandardGangliaReporter

```
{
  "name": "StandardGangliaReporter",
  "type": "org.apache.nifi.reporting.ganglia.StandardGangliaReporter",
  "properties": {
    "Send JVM Metrics": "false",
    "Port": "8649",
    "Hostname": "localhost"
  },
  "propertyDescriptors": {},
  "scheduledState": "ENABLED",
  "schedulingPeriod": "5 mins",
  "schedulingStrategy": "TIMER_DRIVEN",
  "componentType": "REPORTING_TASK"
}
```

Note:

If you plan to use more than one reporting task with the same class name in the deployment, add an optional "bundle" section to the JSON file.

```
  {
    "name": "StandardGangliaReporter",
    "type": "org.apache.nifi.reporting.ganglia.StandardGangliaReport
er",
    "bundle": {
      "group": "org.apache.nifi",
      "artifact": "nifi-standard-nar",
      "version": "1.24.0.2.3.12.0-17"
    },
    "properties": {
      "Send JVM Metrics": "false",
      "Port": "8649",
      "Hostname": "localhost"
    },
    "propertyDescriptors": {},
    "scheduledState": "ENABLED",
    "schedulingPeriod": "5 mins",
    "schedulingStrategy": "TIMER_DRIVEN",
    "componentType": "REPORTING_TASK"
  }
```

For more information, see StandardGangliaReporter in the CDF documentation.