Cloudera Data Warehouse on premises 1.5.5

Supporting Cloudera Data Warehouse

Date published: 2020-08-17
Date modified: 2025-11-08

https://docs.cloudera.com/

Legal Notice

Contents

Monitoring Cloudera Data Warehouse service resources with Grafana dashboards

Grafana is visualization and analytics software that enables the development of dashboards to monitor metrics data. You can access pre-built Grafana dashboards to monitor Virtual Warehouses and your compute cluster in Cloudera Data Warehouse.

You connect to prebuilt dashboards to view metrics of Cloudera Data Warehouse operations. Cloudera provides prebuilt Grafana dashboards for Hive, Impala, and Hue dashboards of metrics data, charts, and other visuals.

Using Grafana, Cloudera metrics are centralized in a single spot, stored in the Prometheus database, and monitored by Prometheus. Your workload databases are not involved in any way. You can immediately view the following pre-built dashboards:

Hive dashboards

The Hive dashboards cover the following operations of the Hive SQL engine in Cloudera Data Warehouse:

- Auto-scaling
- Hive metastore
- HiveServer
- The Hive service itself (Hive-Home)
- LLAP

Impala dashboards

The Impala dashboards include the following operations of the Impala SQL engine in Cloudera Data Warehouse:

- Catalog server
- Coordinator
- Executor
- Statestore
- The Impala service itself

The following screenshot shows the available scratch and cache disk utilization graphs for the Impala Virtual Warehouse:

You can view dashboard metrics for different time periods by selecting the period of interest from the time range dropdown in the horizontal navigation.

On the Cloudera Embedded Container Service platform, you can view the CPU, memory, network usage, and disk input-output for each Cloudera Data Warehouse node using the [***ENVIRONMENT-NAME***]-Nodes option. You can also expand the individual dashboards to see more details, as described in the following table:

Dashboard name	Description	Available metrics
CPU	CPU utilization per node	• Usage per node • Usage per user • Usage per system • Idle time • IO wait
Memory	Memory utilization per node	• Usage per node • Buffer cache • Page cache • Total, used, and available
Network	Number of bytes and packets sent and received	• Network transmitted • Network received • Network transmitted by an interface • Network received by an interface
Disk	Disk bytes read and written	• Bytes written • Bytes read • IO wait time

Note: Node-level metrics are currently available only for Cloudera Embedded Container Service environments.

Related Information
Grafana documentation

Connecting to Grafana dashboards in on premises

You can access Grafana from the Cloudera Management Console and view various dashboards related to the Cloudera Data Warehouse service.

Before you begin

You must have an activated Cloudera Data Warehouse environment to view dashboards for Cloudera Data Warehouse service resources in Grafana.

Procedure

1. Log in to the Cloudera Management Console as an Administrator.
2. Go to the **Dashboard** page and click Monitoring Dashboard.
3. In the Grafana web interface, click  in the left navigation menu, and select Manage.

   A list of available monitoring dashboards is displayed:

CDW service area	Dashboard topics
CDP Control Plane	Alerts generated by Cloudera Management Console, pod status, including count, restarts, CPU usage, memory usage, and container memory and CPU usage
Data Warehouse compute auto-scaling	Auto-scaling
Hive	Hive MetaStore (HMS), HiveServer2, Hive service (Hive-Home), and several dashboards for LLAP
Impala	Impala components: catalog server, coordinator, executor, statestore, and the overall Impala service (Impala-Home)

CDW service area	Dashboard topics
Hue	overall Hue service (Hue-Home)
Overview	Kubernetes alerts, pod status, pod CPU usage, pod memory usage, app CPU usage, app memory usage, container memory usage, container CPU usage
Nodes	CPU, memory, network usage and disk IO metrics at the node level for a given environment

Limitations of Grafana in Cloudera Data Warehouse on premises

Learn about the Grafana capabilities in Cloudera Data Warehouse that Cloudera does not support. Grafana in Cloudera Data Warehouse is intended for use by cluster operations professionals who are familiar with monitoring tools, interpreting metrics, and performing maintenance.

Supported features

- Viewing and organizing Grafana dashboards.

Unsupported features

Storing metrics longer than 15 days, or consuming more than 90GB of disk space is not supported. Metrics older than 15 days are deleted. If the stored metrics consume more than 90GB of disk space, metrics will be deleted regardless of the number of days stored.

Forwarding logs to your observability system

You can forward logs from environments activated in Cloudera Data Warehouse to observability and monitoring systems such as Datadog, New Relic, or Splunk. You learn how to configure a Cloudera Data Warehouse environment for these systems.

About this task

After configuring log forwarding as described in this task, logs flow from Cloudera Data Warehouse to your system automatically. You enjoy the convenience of sorting, searching, and viewing logs on your own system instead of grepping logs from diagnostic bundles on S3 or ABFS. In addition to configuring the log forwarding, you configure removal of debug logs and text strings from the logs. You can configure log forwarding to one of the following observability systems:

- Datadog — https://github.com/DataDog/fluent-plugin-datadog
- Honeycomb.io — https://docs.honeycomb.io/getting-data-in/logs/log-collectors/fluentd/
- New Relic — https://github.com/newrelic/newrelic-fluentd-output
- Splunk — https://github.com/splunk/fluent-plugin-splunk-hec (covers both Splunk-HEC and Splunk-SCS)

You create the log forwarding configuration in valid fluentd format. The configuration is inserted into a larger fluentd configuration. All fluentd events are copied and relabeled with the new label @cloudera_cdw. Your custom configuration is then inserted between <label> tags:

```
<label @cloudera_cdw>
```

customer config goes here

```
</label>
```

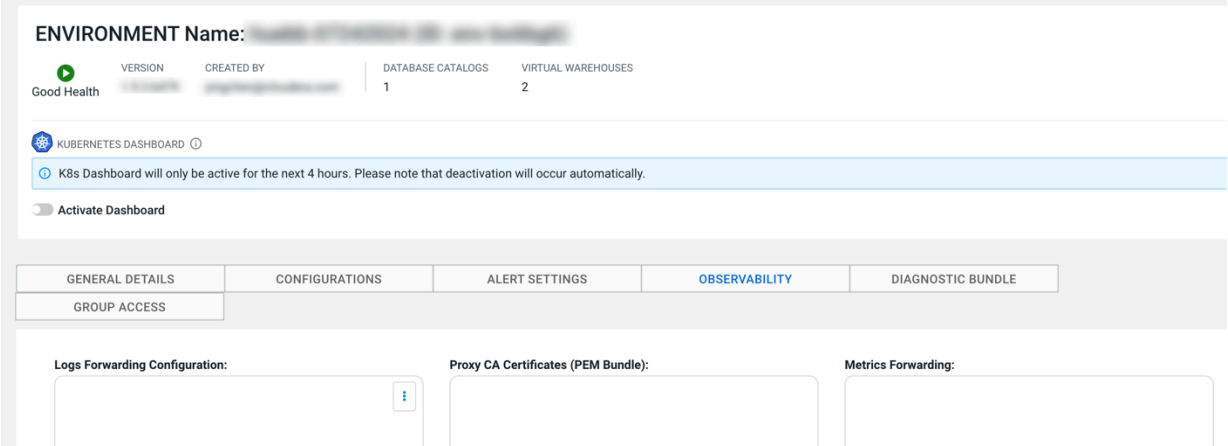
You can use any of the built-in fluentd filter, formatter, parser, or output plugins to build the custom config.

Before you begin

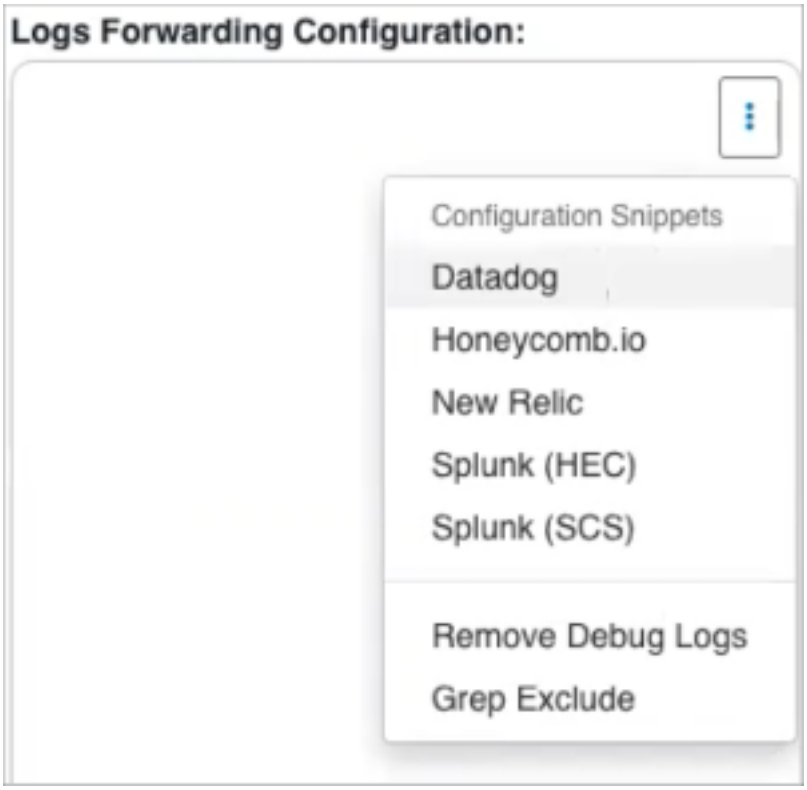
- Before configuring log forwarding you must activate an AWS environment or activate an Azure environment in Cloudera Data Warehouse.
- You must be familiar with fluentd and accept the responsibility of configuring log forwarding to your observability systems.

Procedure

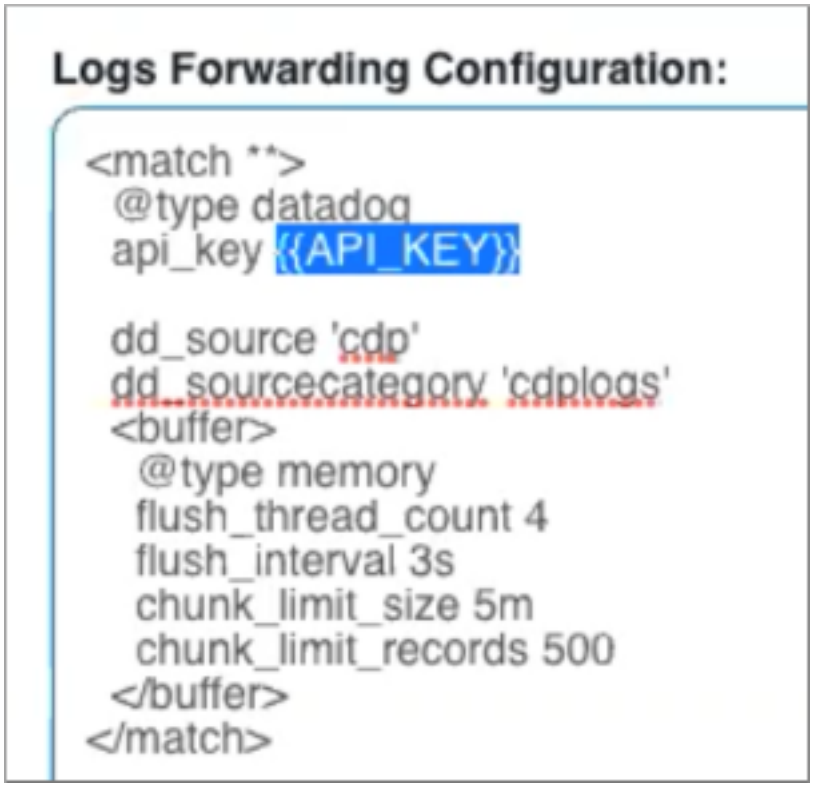
1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate your environment, and click ⋮ Edit Observability .



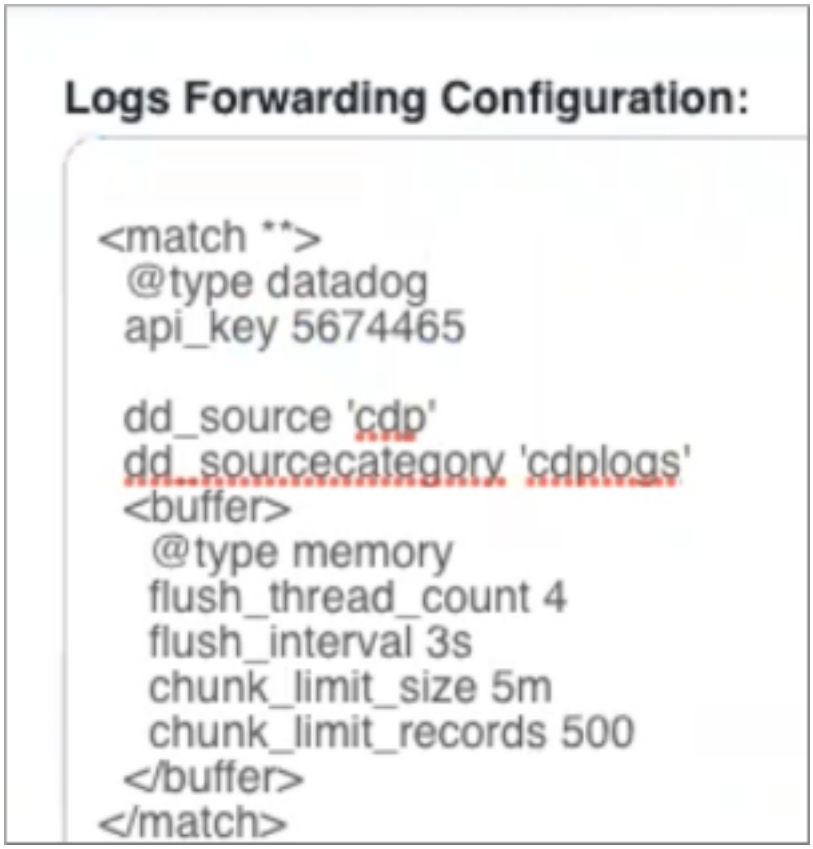
3. Decide how you want to create the fluentd config.
   - Write your own fluentd config from the ground up.
   - Use a Cloudera-provided snippet as a template to write your fluentd config.
4. In Log Forwarding Configuration, click ⋮ .



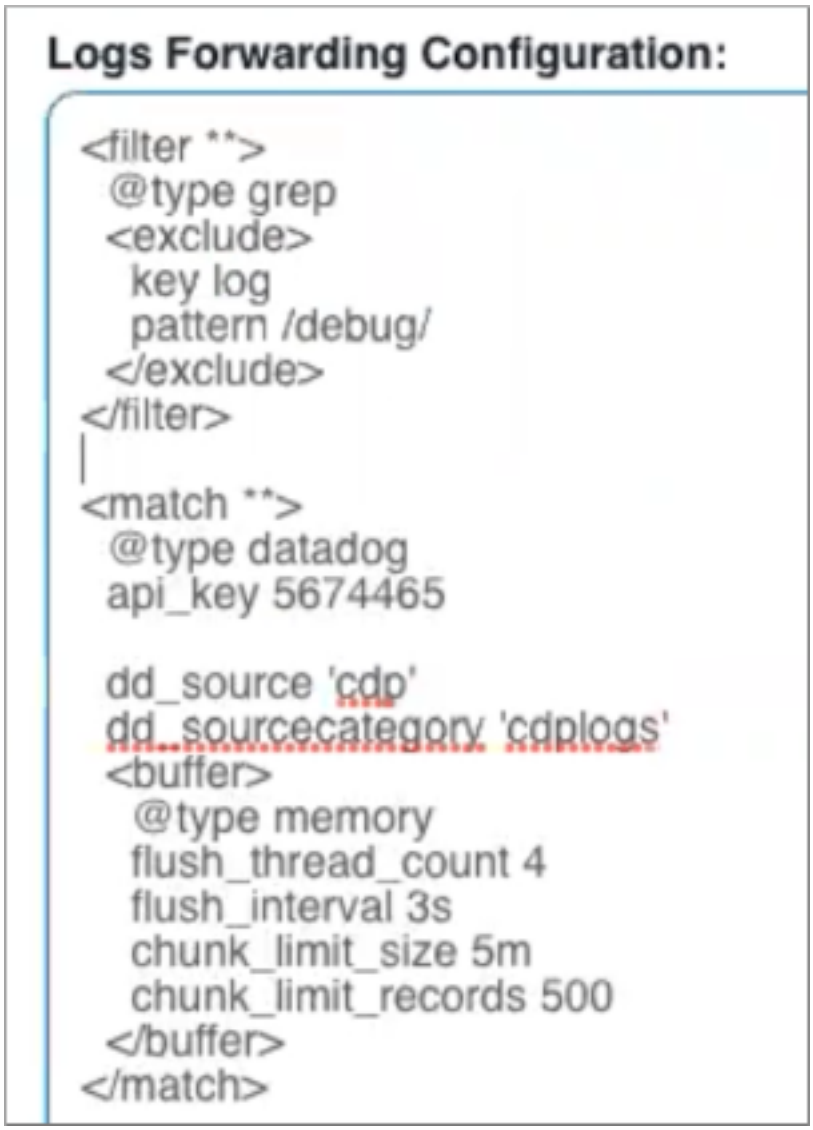
5. Select one of the systems, such as Datadog, to configure.
   A fluentd snippet appears. For example, the Datadog snippet appears:



6. Replace the snippet with the fluentd config you wrote from the ground up, or customize the provided snippet.
   For example, to customize the provided snippet replace the placeholder {{API Key}} with the actual key.



7. (Optional) If debug level log messages are not desired, add a fluentd filter to remove them: In the environment, click ⋮, and select Remove Debug Logs.
   The fluentd snippet appears for removing debug logs. For example:



   No user customization is necessary to remove debug logs.

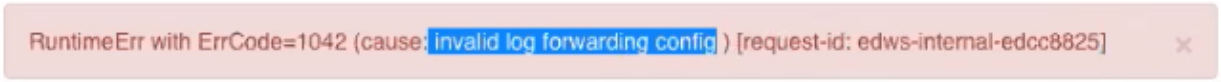
8. (Optional) If certain log messages do not provide value for you, remove them with a fluentd grep exclude filter:

   In the environment, click ⋮, select Grep Exclude, and replace {{PATTERN}} with the grep expression that matches the phrase you want to exclude.

   For more information about using Grep Exclude, see https://docs.fluentd.org/filter/grep.

9. If you use a proxy server for outbound traffic, provide the proxy server's CA certificates PEM bundle as described in the next task.

10. Click Apply Changes.
    Cloudera Data Warehouse tests the log forwarding configuration and proxy CA certificates bundle, and saves the configuration if both are valid. An invalid log forwarding config error message appears in the event of a configuration problem. For example:

    If your configuration is valid, Cloudera Data Warehouse initiates a restart of fluentd to apply the updated config. You see the following indicators of success:

    - The environment Running indicator changes, blinks Updating, and then once again says Running.
    - You see logs appearing in your observability system.

      Many factors affect how long it takes for forwarding to begin, but generally, the bigger your Cloudera Data Warehouse environment, the longer it takes.

Providing proxy CA certificates

If you use a TLS-terminating proxy server to inspect outbound internet traffic, you need to provide the proxy server's CA certificates bundle in PEM bundle format when you configure log forwarding.

About this task

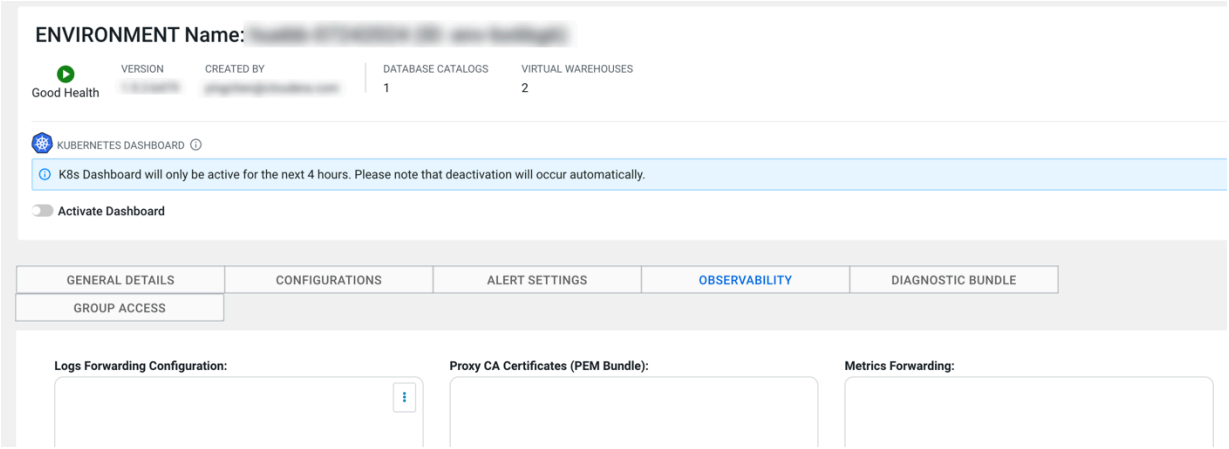
You learn how to use the Observability tab in Cloudera Data Warehouse Environment Details to configure the Proxy CA Certificates (PEM Bundle) field.

Before you begin

Before you apply the proxy CA certificate to a configuration of log forwarding, you must provide a configuration in the Logs Forwarding Configuration section of the Observability tab.

Procedure

1. In the Cloudera Data Warehouse service, go to the Environments tab.
2. Locate your environment and click ⋮ Edit Observability .
   Environment details include a UI for configuring log forwarding.

3. Obtain and copy your proxy server's CA certificates PEM bundle.
4. In Proxy CA Certificates (PEM Bundle), paste the copy of the PEM bundle.

5. Click Apply Changes.

   If the certificate and log forwarding configuration are valid, log forwarding begins. If the certificates are invalid, an error message occurs.

   The log forwarding configuration and certificates are not saved.

OpenTelemetry support in Cloudera Data Warehouse

Learn how OpenTelemetry (OTel) improves query performance visibility and supports Hive and Impala in Cloudera Data Warehouse

OpenTelemetry (OTel) provides an open-source solution for collecting, processing, and exporting telemetry data, including metrics from applications. OTel helps users gain visibility into query performance and troubleshoot query failures. OpenTelemetry (OTel) is supported with Hive and Impala in Cloudera Data Warehouse.

Related Information

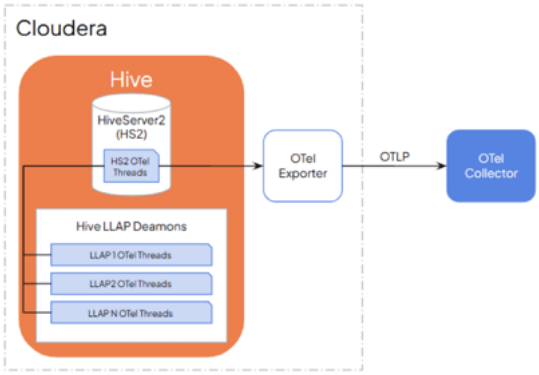
OpenTelemetry

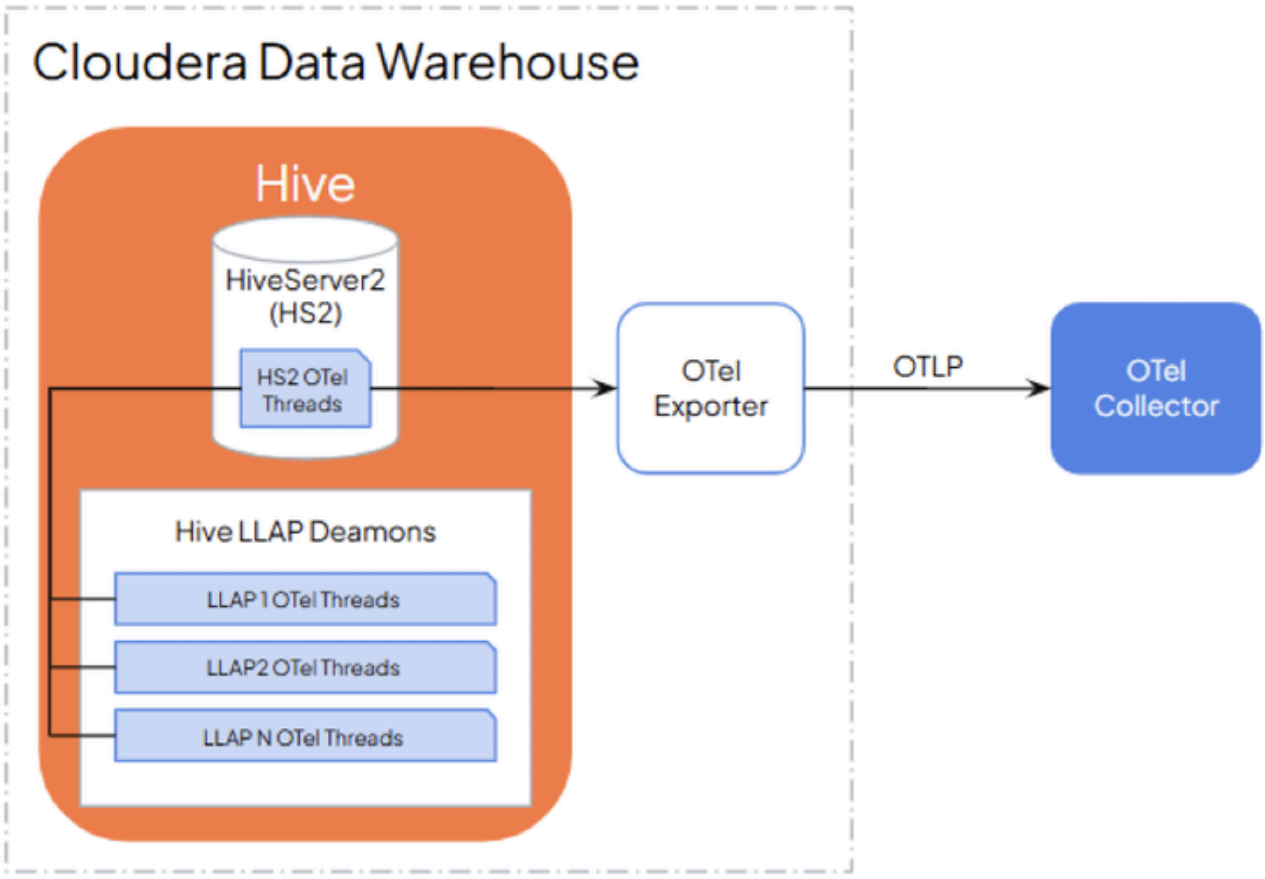
OpenTelemetry support for Hive

Learn how Hive uses OpenTelemetry (OTel) to collect and publish telemetry data, processed by a user-configured OTel collector for visualization in systems like Jaeger and Prometheus.

Overview

As part of this offering, Hive in Cloudera Data Warehouse includes an OTel exporter that helps to collect, filter, and publish telemetry information, such as infrastructure and workload metrics, live and historical query data.





HiveServer2 and LLAP each transmit telemetry data to an OTel agent. The OTel agent independently transmits the data to a customer configured OTel collector instance for processing. The processed data can be exported and visualized through backend systems, such as Jaeger, Zipkin, Prometheus.

Note: The OTel collector is not part of the Cloudera Data Warehouse deployment. You must have your own instance of the OTel collector that is configured with backend instances (Jaeger, Zipkin, Prometheus).

HiveServer2 and LLAP integration with OTel

OTel threads function independently in both HiveServer2 and LLAP daemons. When query execution begins, these threads capture essential telemetry data, which is transmitted to an OTel collector based on a configurable recurring schedule. The collected data is then processed and forwarded to backend systems for visualization.

HiveServer2 integrates with the OTel agent to expose both query-related data and JVM metrics. A dedicated thread or service runs within HiveServer2 to handle this integration. This thread collects query details and metrics, which are then transmitted through the OTel agent. These transmitted metrics can be collected by OTel collectors for analysis and visualization.

Metrics specific to each LLAP daemon, such as JVM and memory-related statistics, are also transmitted for detailed observability.

Availability

OTel support for Hive is made effective as of the Cloudera Data Warehouse on premises1.5.5 version. After upgrading the Cloudera Data Warehouse version, you must also upgrade existing Hive Virtual Warehouses to enable and configure the OTel integration

Benefits of OTel Hive Integration

This topic explains the benefits of integrating OpenTelemetry (OTel) with HiveServer2 and LLAP, focusing on enhanced observability, telemetry data insights, and optimized performance.

The integration of OpenTelemetry (OTel) with HiveServer2 and LLAP provides advanced telemetry capabilities, enabling better observability and diagnostics while maintaining optimal system performance.

Telemetry Data Exposed

OTel integration allows HiveServer2 and LLAP to expose the following telemetry data through an OTel collector:

- Metrics: Infrastructure and workload metrics, such as JVM memory usage, thread counts, and Operating System-related insights.
- Live Query Data: Tracking of active query lifecycle events, including execution times, stages, and error messages.
- Historical Query Data: Detailed query execution metadata for analysis and diagnostics.

Performance Benefits

The OTel integration is designed to enhance observability while ensuring HiveServer2 and LLAP maintain optimal performance.

Optimized for Minimal Impact: Integrating OTel with HiveServer2 and LLAP ensures seamless performance. By utilizing an independent thread or service to collect, translate, and expose metrics already tracked by HiveServer2, the integration ensures query execution remains smooth, without any noticeable delays or disruptions.

Scalable for Future Metrics: The system is well-prepared to handle new metrics if introduced. While tracking additional metrics may require some extra resources, this scalable design ensures the system remains efficient, with any potential impact being manageable and dependent on your specific use case.

Efficient Memory Usage: The OTel thread is optimized for minimal memory consumption, efficiently managing the receive, translate, and expose phases. The memory usage is small and well within acceptable limits, ensuring it doesn't affect overall system performance, even during extended operations.

Configuring OTel in HiveServer2

Learn how you can enable OTel in HiveServer2 and configure certain properties that will enable you to optimize the data collection.

Before you begin

Ensure that you are on Cloudera Data Warehouse on premises 1.5.5 or higher version.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
   The Overview page is displayed.
2. From the Overview page, click the Virtual Warehouses tab, identify the Hive Virtual Warehouse that you want to configure, and then click  to edit.
3. In the Virtual Warehouse details page, click Configurations Hiveserver2 and then select hive-site from the Configuration files drop-down.

4. Search for hive.otel and modify the values as required:
   a) hive.otel.metrics.frequency.seconds (Default: 0s)

      Specifies the frequency at which telemetry data is transmitted to the OTel collector. By default, the value is set to 0 seconds indicating that OpenTelemetry data collection is disabled. Enter a value greater than 0 to enable OpenTelemetry.

      If the value is 5s. This indicates that telemetry data is transmitted to the OTel collector every 5 seconds.
   b) hive.otel.collector.endpoint

      Specifies the endpoint where all the OpenTelemetry Protocol (OTLP) traces and metrics are transmitted. The endpoint represents the address of an OTel collector. The endpoint must be a valid URL with https scheme.

      https://<otel-collector-host>:<port>

      Important: Cloudera recommends using https to ensure secure data transmission.
   c) hive.otel.exporter.timeout (Default: 10m)

      Specifies the maximum time allowed for the OTel agent to complete a transmit operation. The transmit operation times out if it exceeds the specified time.
   d) hive.otel.retry.initial.backoff (Default: 10s)

      Specifies the initial time delay before attempting to retry a failed transmit operation. The value serves as the starting point for the exponential backoff strategy.
   e) hive.otel.retry.max.backoff (Default: 1m)

      Specifies the maximum time that the OTel agent should wait between retries. This sets an upper limit on the backoff interval ensuring that retry export operations do not exceed the specified duration even with exponential backoff.
   f) hive.otel.retry.backoff.multiplier (Default: 5f)

      Specifies the factor by which the retry interval increases after every failed attempt. This determines how much the backoff interval increases after each failed attempt, following an exponential backoff strategy.
5. Click Apply Changes and restart the Hive Virtual Warehouse.

Telemetry data exposed to OTel collector for hive

HiveServer2 transmits telemetry data related to live queries, completed queries, task-level details, JVM metrics related to memory usage and thread count, and Operating System level statistics.

Query and Task related insights

Live queries

The following metrics related to live HiveServer2 queries are transmitted to the OTel collector:

Metrics	Description
QueryId	Represents the unique identifier for the query.
QueryString	The SQL statement of the query.
UserName	The user who submitted the query.
ExecutionEngine	The query execution engine used to process the query, typically Tez.
ErrorMessage	Errors encountered during query execution.

Completed queries

The following metrics related to completed HiveServer2 queries are transmitted to the OTel collector:

Metrics	Description
QueryId	Represents the unique identifier for the query.

Metrics	Description
QueryStartTime	Timestamp when the query execution started.
EndTime	Timestamp when the query execution completed.
OperationId	Unique identifier for tasks related to query execution. For example, a0fe8acc-6b9a-4f54-8537-f7b3bf7dea72
OperationLogLocation	Path to the local log file for additional query log entries.
ErrorMessage	Errors encountered during query execution.
ExplainPlan	Output showing how the engine executed the query.
FullLogLocation	Location of the complete application logs related to the query.
Running	Indicates whether the query is currently in progress.
Runtime	Duration of the query execution in seconds or minutes.
UserName	The user who submitted the query.
ExecutionEngine	The query execution engine used to process the query, typically Tez.
State	Current state of the query (e.g., RUNNING, SUCCESS, ABORTED, or FAILED).
SessionId	Identifier for the session in which the query was executed.

Task-level details

The following metrics related to tasks are transmitted to the OTel collector:

Metrics	Description
TaskId	Unique identifier for each task.
Name	Task types such as MAPRED (Mapper/Reducer tasks), DEPENDENCY_COLLECTION, or STATS TASK. If multiple tasks exist, integers are appended.
TaskType	Representation with unique values while execution such as MAPRED, DEPENDENCY_COLLECTION, MOVE, or STATS.
Status	Current task status (e.g. Success).
StatusMessage	Status with detailed task information
ExternalHandle	DAGID, used for query optimization and execution.
ErrorMsg	Error details if the task failed.
ReturnValue	Task result. A value of 0 indicates success, while negative integers indicate an issue.
BeginTime	Time when the task execution started.
ElapsedTime	Time spent executing the task.
EndTime	Time when the task execution finished.

Java Virtual Machine (JVM) metrics

HiveServer2 collects various JVM metrics, including:

- Memory usage: Data such as heap and non-heap memory usage.
- Thread count: Counts of threads in different states (for example, runnable, waiting).
- OS-Level statistics: CPU load, memory size, and swap space details.

Metrics related to memory usage

The following metrics related to JVM heap and non-heap memory usage are transmitted to the OTEL collector:

Metrics	Description
memNonHeapUsedMGauge	Size (in MB) of non-heap memory currently used by the JVM.
memNonHeapCommittedM	Amount of non-heap memory (in MB) reserved by the JVM for internal use.
memNonHeapMaxM	Maximum allowable size (in MB) for non-heap memory.
memHeapUsedM	Size (in MB) of heap memory currently used by the JVM.
memHeapCommittedM	Size (in MB) of heap memory reserved by the JVM.
memHeapMaxM	Maximum allowable size (in MB) for heap memory.
memHeapMaxM	Maximum memory available (in MB) to the JVM.

Metrics related to thread count

The following metrics related to the JVM threads in different states (runnable, waiting) are transmitted to the OTel collector:

Metrics	Description
threadsNew	Number of threads currently in the "NEW" state within a Java application.
threadsRunnable	Number of threads ready to run or currently executing on the CPU.
threadsBlocked	Number of threads waiting to acquire a lock.
threadsWaiting	Number of threads waiting indefinitely for a signal from another thread.
threadsTimedWaiting	Number of threads waiting for a specific duration before proceeding.
threadsTerminated	Number of threads where execution is completed.

OS-level statistics

The following OS related metrics, such as CPU load, memory size, swap space details are transmitted to the OTel collector:

Note: These metrics are transmitted only for Unix-based operating systems.

Metrics	Description
systemCpuLoad	Measures the CPU load of the host machine, container, or pod.
committedVirtualMemorySize	Amount of allocated memory, including both physical RAM and virtual memory reserved by the operating system.
processCpuTime	Total CPU time consumed by a specific process or thread.
freePhysicalMemorySize	Amount of unused physical memory (RAM) available to processes and the operating system, container, or pod.
freeSwapSpaceSize	Amount of swap space currently available.
totalPhysicalMemorySize	Total installed physical memory (RAM) in the system.

Metrics	Description
processCpuLoad	Percentage of CPU load consumed by a specific process.

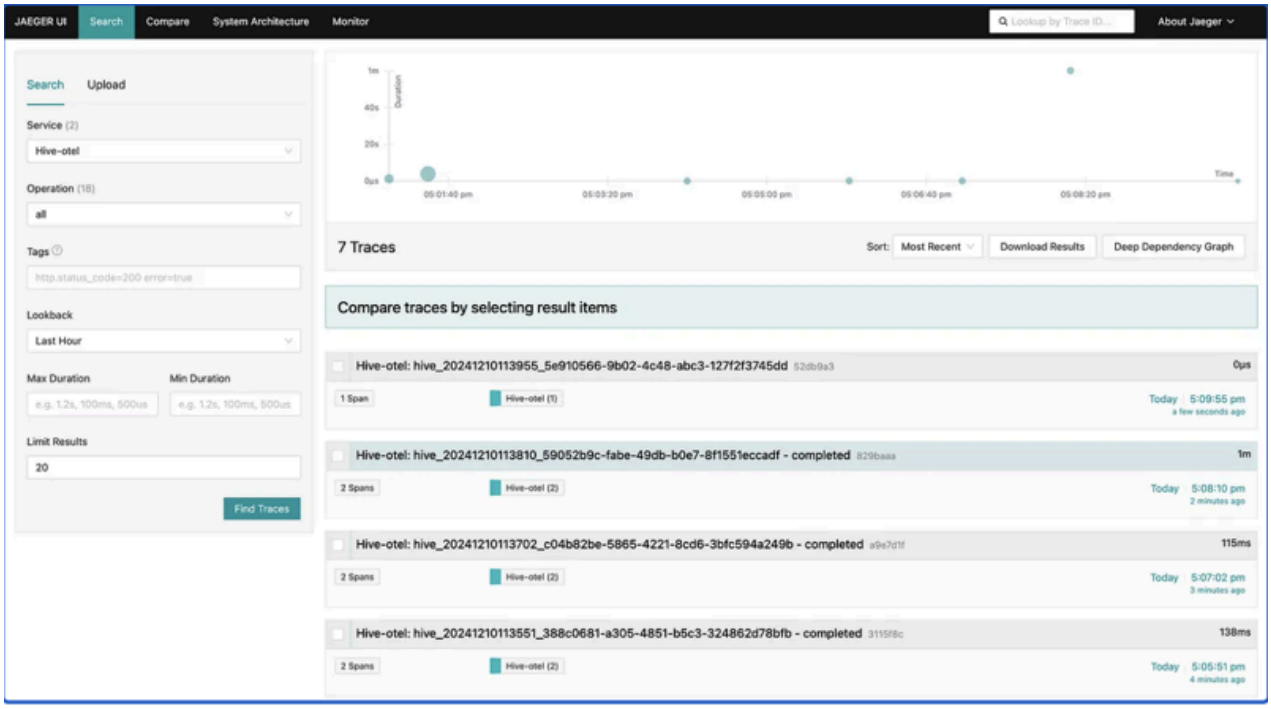
Example Visualizations through OTel backend

This section provides some examples of OTel backend systems that allow you to view the telemetry data that is processed and exported from an OTel collector.

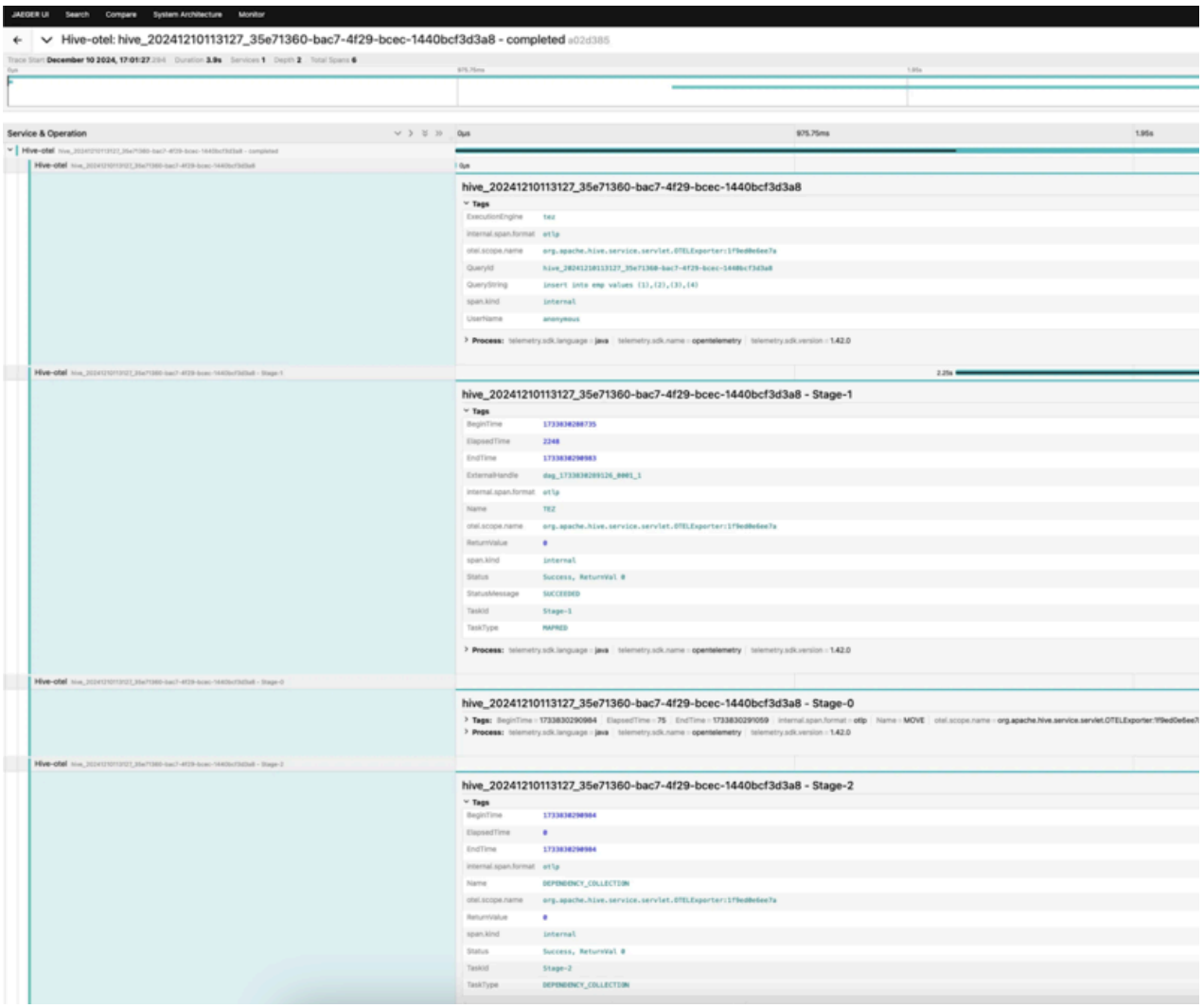
Important: The example backend systems shown here are just for representation. Along with having your own instance of the OTel collector, you must also have your own instances of Jaeger, Zipkin, Prometheus and Grafana to view the telemetry data.

Visualizing through Jaeger

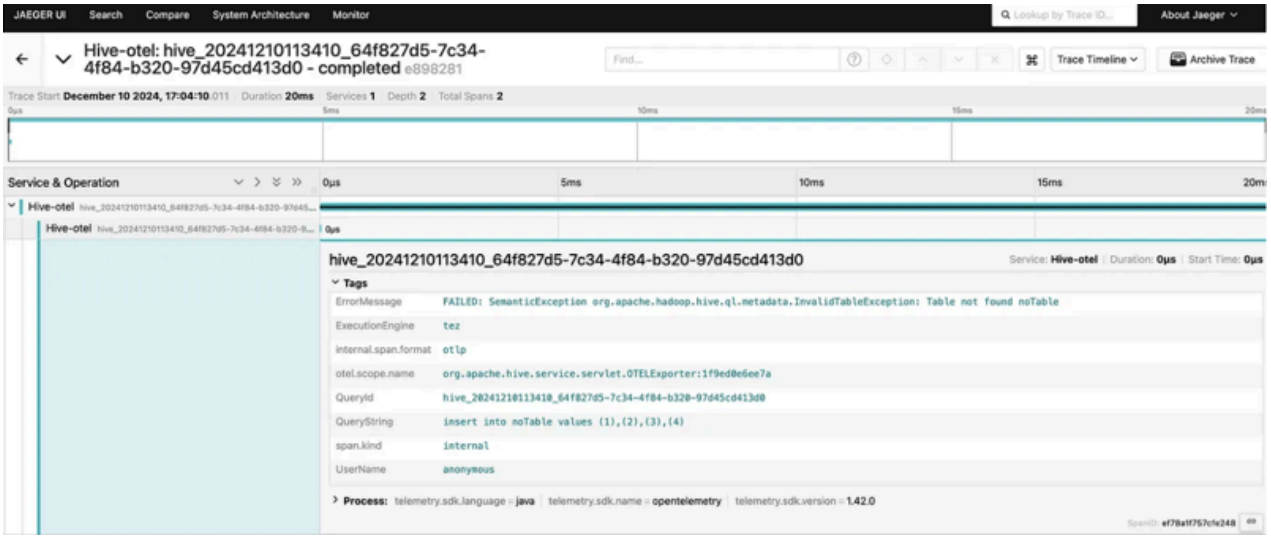
The following image represents a Jaeger UI displaying traces for 4 queries out of which 1 is a live query and the remaining 3 are completed queries. The completed queries are denoted by the completed suffix and the running queries are denoted by just the Query ID and do not have a suffix.



You can click on a trace to view more details about the attributes and tasks. This helps you understand task breakdowns and performance aspects of individual query components.



Failed queries display an ErrorMessage in the expanded view enabling you to troubleshoot and debug effectively.



Limitation of OpenTelemetry support for Hive

Learn about the current limitations of OTel integration, including its focus on metrics and events, while future-proofing for logs. Understand the fixed nature of event data and its implications for query observability.

Telemetry Data Scope

The scope of telemetry data is currently limited to only metrics and events. Logs and traces are under consideration for a future release.

Fixed Event Data

The events being sent through an OTel agent are not configurable and can include Personally Identifiable Information (PII) within the SQL statement of the query.

OpenTelemetry support for Impala

The Impala OpenTelemetry integration enables real-time query observability and centralized telemetry data collection, including lifecycle events and resource usage.

Overview

Impala telemetry data is integrated with OTel-compatible collectors. This provides a centralized flow of live query insights, with SELECT queries represented as OTel traces, and reduces the friction of sourcing data from multiple places.

Impala integration with OTel

Impala integrates the OTel C++ SDK to emit query lifecycle data. The system already tracks specific phases and events for each query and records them in the query profile timeline section. By emitting these events to an OTel collector, observability systems can track active queries in near real-time.

Collected telemetry data

Telemetry data emitted from Impala carries crucial information that is currently available only in the query profile and workload management tables. Telemetry data includes the following data:

1. The initiating user
2. The SQL statement
3. Memory estimates and actual use
4. Other important data related to the query lifecycle

Availability

OTel support for Impala is made effective as of the Cloudera Data Warehouse on premises 1.5.5 SP1 version. After upgrading the Cloudera Data Warehouse version, you must also upgrade existing Impala Virtual Warehouses to enable and configure the OTel integration.

Benefits of OTel Impala Integration

Learn about the benefits of integrating OpenTelemetry (OTel) with Impala focusing on enhanced observability, telemetry data insights, and optimized performance.

Enhanced query lifecycle and historical data

Integrating OTel with Impala provides enhanced observability through comprehensive data collection. The following data is collected:

- Live query data – Data about important events in the lifecycle of actively running queries is sent to collectors in near real-time as the events happen.
- Historical query data – Data about completed queries can be retained by the destination OTel trace management system.

Performance and scalability impact

The integration is designed to have negligible impact on Impala's performance and is built for scalability. The following performance and scalability impacts are valid for the integration:

- The performance impact on Impala is negligible because the system already collects all the necessary event and metric data.
- The process of sending data to the OTel endpoint is handled out-of-band in a separate thread, limiting any performance impact to just the sending of this data.
- Scalability concerns are primarily limited to the OTel Collector endpoint. Impala does not encounter scalability issues as long as communication with the Collector happens without delay.

Configuring OTel in Impala

Learn how you can enable OTel in Impala and configure certain properties that enable you to optimize the data collection.

Before you begin

- You must ensure that you are on Cloudera Data Warehouse on premises 1.5.5 SP1 or higher version.

Procedure

1. Log in to the Cloudera web interface and navigate to the Cloudera Data Warehouse service.
   The **Overview** page is displayed.
2. From the **Overview** page, click the Virtual Warehouses tab, identify the Impala Virtual Warehouse that you want to configure, and then click the  icon and choose the Edit action from the list of actions.
3. In the **Virtual Warehouse Details** page, go to Configurations Impala Coordinator and then select the flagfile option from the Configuration files list.

4. Click the Add Custom Configuration button and manually add the configurations as needed.
   a) Add the --otel_trace_collector_url configuration key.

      Specifies the URL of the OpenTelemetry collector to which trace data will be exported. The endpoint must be a valid URL.

      ```
      --otel_collector_url= https://collector-endpoint
      ```

      Note: For secure data transmission, Cloudera recommends using https in the URL.

   b) Add the --otel_trace_enabled configuration key. The default value is false.

      If set to true, OpenTelemetry traces will be generated and exported to the configured OpenTelemetry collector.

   c) Add the otel_trace_batch_max_batch_size configuration key. The default value is 512.

      Specifies the maximum batch size of every export to the OTel Collector. This configuration is applicable when the value of the otel_trace_span_processor configuration is batch.

   d) Add the otel_trace_batch_queue_size configuration key. The default value is 2048.

      Specifies the maximum buffer or queue size. After the set size is reached, spans are dropped. This configuration is applicable when the value of the otel_trace_span_processor configuration is batch.

   e) Add the otel_trace_batch_schedule_delay_ms configuration key. The default value is 5000.

      Specifies the delay interval in milliseconds between two consecutive batch exports. This configuration is applicable when the value of the otel_trace_span_processor is batch.

   f) Add the otel_trace_additional_headers configuration key.

      Specifies a list of additional HTTP headers to be sent with each call to the OTel Collector .

   g) Add the otel_trace_ca_cert_path configuration key.

      Specifies the path to a file containing a CA certificates bundle.

   h) Add the otel_trace_ca_cert_string configuration key.

      Specifies a string containing a CA certificates bundle.

   i) Add the otel_trace_compression configuration key. The default value is true.

      If set to true, uses ZLib compression for sending data to the OTel Collector.

   j) Add the otel_trace_timeout_s configuration key. The default value is 10.

      Specifies the export timeout in seconds.

   k) Add the otel_trace_tls_insecure_skip_verify configuration key. The default value is false.

      If set to true, skips verification of the collector TLS certificate.

      Note: Cloudera recommends setting this configuration to false only for development or testing purposes.

   l) Add the otel_trace_tls_minimum_version configuration key. The default value is the overall minimum TLS version.

      Specifies the minimum allowed TLS version.

   m) Add the ssl_minimum_version configuration key. The default value is tlsv1.2.

      Specifies the minimum SSL or TLS version that Impala Thrift services are expected to use for both client and server connections. This flag applies to all Impala Thrift services, and supported versions include TLSv1, TLSv1.1, and TLSv1.2.

      Important: The value of this flag is used if the otel_trace_tls_minimum_version configuration key is not specified.

   n) Add the otel_trace_ssl_ciphers configuration key. The default value is the value of Impala ssl_cipher_list startup flag.

      Specifies a list of allowed TLS cipher suites when using TLS 1.2.

   o) Add the otel_trace_tls_cipher_suites configuration key. The default value is the value of Impala tls_ciphersuites startup flag.

Specifies a list of allowed TLS cipher suites when using TLS 1.3.

p) Add the otel_trace_retry_policy_max_attempts configuration key. The default value is 5.

Specifies the maximum number of call attempts, including the original attempt.

q) Add the otel_trace_retry_policy_initial_backoff_s configuration key. The default value is 1.

Specifies the initial backoff delay between retry attempts in seconds.

r) Add the otel_trace_retry_policy_max_backoff_s configuration key. The default value is 0.

Specifies the maximum backoff delay between retry attempts in seconds. A value of 0 or less indicates that the configuration key is not set.

s) Add the otel_trace_retry_policy_backoff_multiplier configuration key. The default value is 2.

Specifies the factor by which the retry interval increases after every failed attempt.

t) Add the otel_debug configuration key. The default value is false.

If set to true, this outputs additional debug information.

5. Click the Apply Changes button and restart the Impala Virtual Warehouse.

Telemetry data exposed to OTel collector for Impala

OpenTelemetry (OTel) provides an open-source solution for collecting, processing, and exporting telemetry data. In Impala, query lifecycle data is structured as OTel traces, with a root span and multiple child spans. The following tables detail the attributes and events associated with these traces.

Root Span of the Trace

The root span represents the entire query and is of kind SERVER.

Metrics	Type	Description	Example
ClusterId	String	A string that uniquely identifies a cluster, determined by the value of the --cluster_id startup flag.	impala-1982661901-6j1z
EndTime	Millisecond epoch time	The time when the query finished in millisecond epoch time.	1743474118301
ErrorMessage	String	A string containing the error message received from the query execution, or an empty string if the query completed successfully.	Invalid syntax
OriginalQueryId	String	When a query is retried, a string containing the Impala query ID of the original query. Otherwise, an empty string.	ef403b2690d243be:b960c0ba00000000
QueryId	String	The Impala query ID.	ef403b2690d243be:b960c0ba00000000
QueryStartTime	Millisecond epoch time	The time when the query was first received in millisecond epoch time.	1743473803667
RequestPool	String	A string containing the name of the request pool the query will be scheduled into.	default-pool
RetriedQueryId	String	The ID of the query that successfully retried this query.	3a43a57ad0bf36df:dcd698b700000000
Runtime	Milliseconds	The time in milliseconds the query ran.	5231
SessionId	String	The Impala session ID.	024f45f3e0019fed:eb0fe00c00000000
State	String	The query state.	FINISHED, EXCEPTION, or RETRIED

Metrics	Type	Description	Example
QueryType	String	The type of the statement in uppercase letters, such as QUERY, DML, or DDL.	DML
UserName	String	The user who submitted the query.	usr123

Child spans

Child spans are of kind INTERNAL and contain both global and specific attributes and events.

Global child span attributes

The global child span attributes are present on every child span.

Metrics	Type	Description	Example
BeginTime	Millisecond epoch time	The time the span started in millisecond epoch time.	1743473803667
ElapsedTime	Millisecond	The time in milliseconds the span ran.	5231
EndTime	Millisecond epoch time	The time the span finished in millisecond epoch time.	1743474118301
ErrorMsg	String	Error details if a failure occurred during this span.	Could not read file
Name	String	The name of the span, in the {{query_id}} - Query Stage format.	cdf601fa776431c43:5e59cba4fe284ae22 - Submitted
Running	Boolean	A boolean value indicating if the query is actively running and not in planning, admission control, or closing. It is set to false if the query fails during this span.	true
Status	String	The status of the task, for example, OK.	OK

Specific child span attributes

The following table lists the additional attributes for the Init span, which are unique to that stage of the query lifecycle.

Metrics	Type	Description	Example
ClusterId	String	A string that uniquely identifies a cluster, determined by the value of the --cluster_id startup flag.	impala-1982661901-6j1z
DefaultDb	String	The name of the default database.	tpcds
Name	String	The name of the span, in the {{query_id}} - Init format.	cdf601fa776431c43:5e59cba4fe284ae22 - Init
OriginalQueryId	String	The Impala query ID of the original query when retried. Otherwise, an empty string.	ef403b2690d243be:b960c0ba00000000
QueryId	String	The Impala query ID.	ef403b2690d243be:b960c0ba00000000

Metrics	Type	Description	Example
QueryString	String	The redacted string containing the SQL statement.	select * from db.tbl where col1 = “val1”
RequestPool	String	A string containing the name of the request pool the query will be scheduled into.	default-pool
SessionId	String	The Impala session ID.	024f45f3e0019fed:eb0fe00c00000000
UserName	String	The user who submitted the query.	usr123

Query Submitted

Metrics	Type	Description	Example
Name	String	The name of the span, in the {{query_id}} - Init format.	cdf601fa776431c43:5e59cba4fe284ae22 - Submitted

Span: Planning

Metrics	Type	Description	Example
Name	String	The name of the span, in the {{query_id}} - Init format.	cdf601fa776431c43:5e59cba4fe284ae22 - Planning
QueryType	String	The type of the statement in uppercase letters, such as QUERY, DML, or DDL.	DML, QUERY

Admission control

Note: The admission control attributes are only applicable for QUERY and DML statements.

Metrics	Type	Description	Example
Name	String	The name of the span, in the {{query_id}} - AdmissionControl format.	cdf601fa776431c43:5e59cba4fe284ae22 - AdmissionControl
AdmissionResult	String	A string containing the result from admission control.	Admitted immediately
Queued	Boolean	A boolean value where true indicates that the query was queued and false indicates that the query was admitted immediately.	false
RequestPool	String	A string containing the name of the request pool the query will be scheduled into.	default-pool

Query execution

Metrics	Type	Description	Example
Name	String	The name of the span, in the {{query_id}} - Execution format.	cdf601fa776431c43:5e59cba4fe284ae22 - Execution
NumDeletedRows	Integer	An integer containing the total number of rows deleted by the DML statement.	0

Metrics	Type	Description	Example
NumModifiedRows	Integer	An integer containing the total number of rows modified by the DML statement.	0
NumRowsFetched	Integer	An integer containing the total number of rows fetched by the client.	5132

Close

Metrics	Type	Description	Example
Name	String	The name of the span, in the {{query_id}} - Close format.	cdf601fa776431c43:5e59cba4fe284ae22 - Close

OpenTelemetry Impala query tracing example

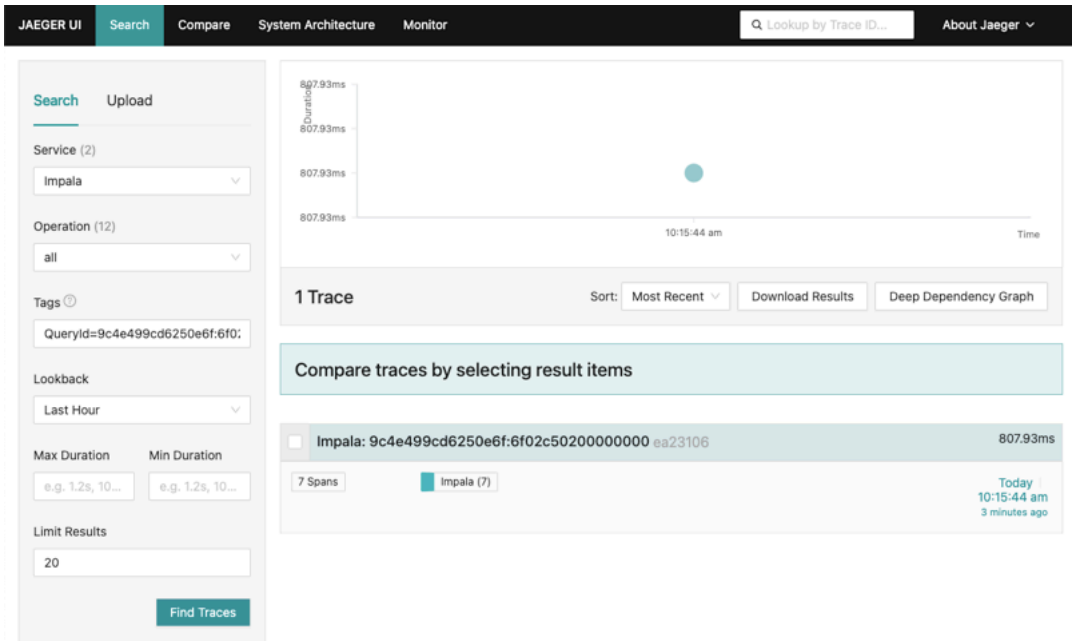
Learn how to use Jaeger to visualize the telemetry data from a simple Impala query.

Important: The example backend systems shown here are just for representation. Along with having your own instance of the OTel collector, you must also have your own instances of Jaeger, Zipkin, Prometheus and Grafana to view the telemetry data.

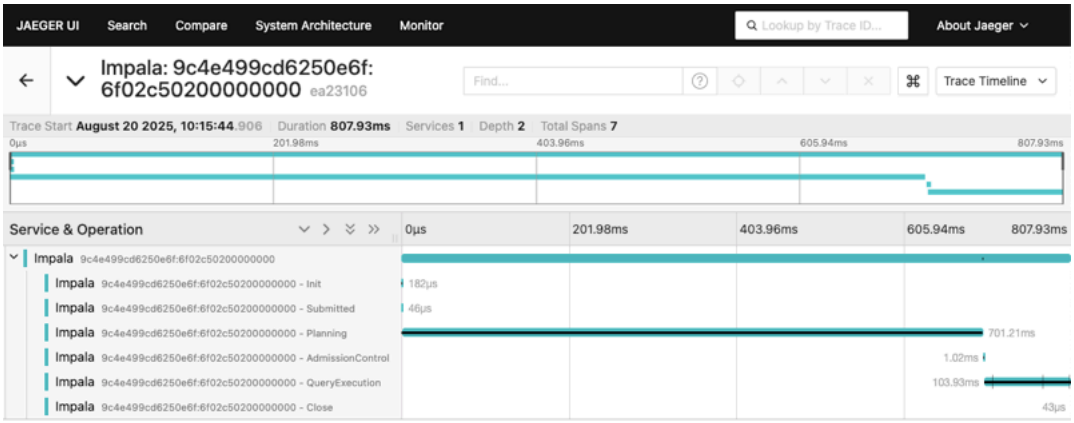
After configuring OpenTelemetry support for your Impala Virtual Warehouse, you can view detailed telemetry data for your queries in a trace visualization system like Jaeger. The following steps show a typical workflow for finding and analyzing a query trace.

Finding a query trace

1. On the Jaeger UI, use the search function to find traces. You can filter by service, such as Impalad, or other tags like the query ID. The search results provide a high-level summary of the traces found.



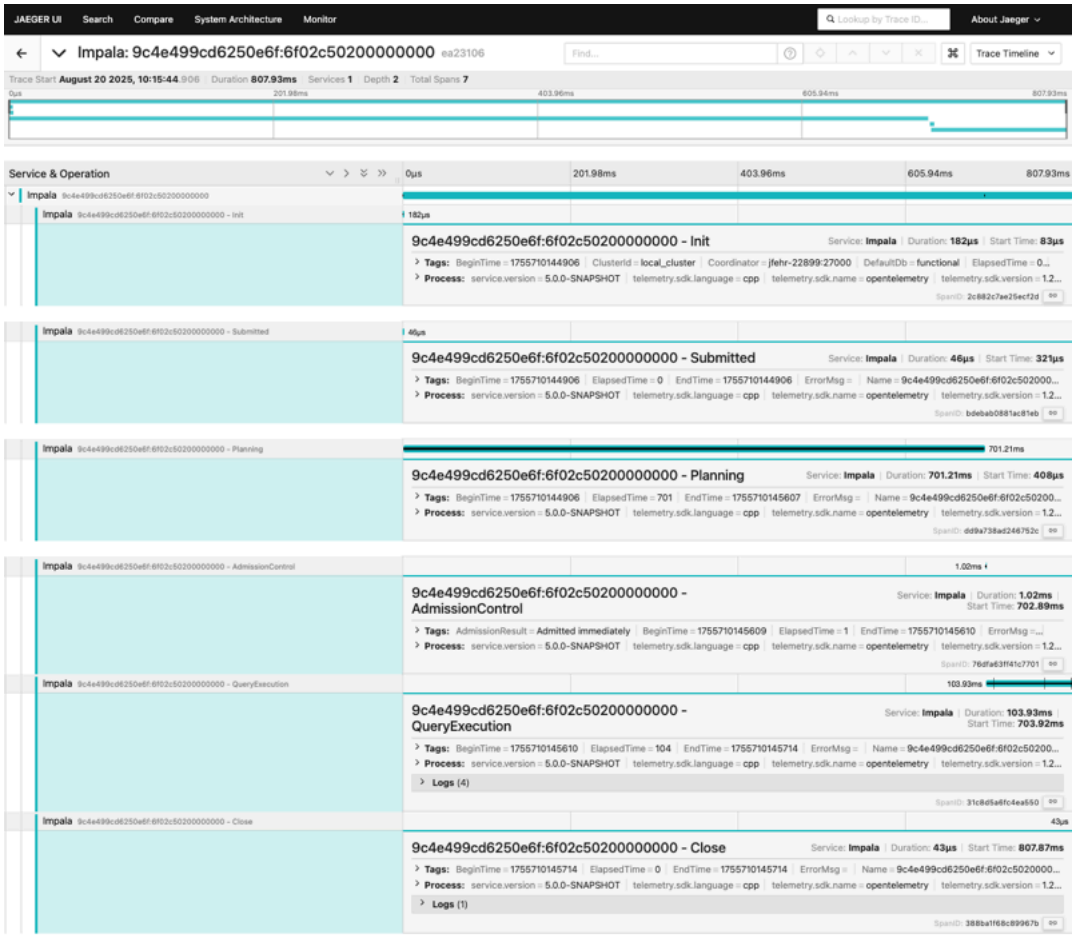
2. From the search results, select a specific trace to view its summary and timeline.



Understanding the query trace timeline

A trace provides a detailed breakdown of a query execution from start to finish. Each segment in the timeline is a span, representing a specific operation. The main query is represented by a root span, and its various stages are shown as child spans.

The following image shows the full timeline of a simple query, with a breakdown of its core stages.

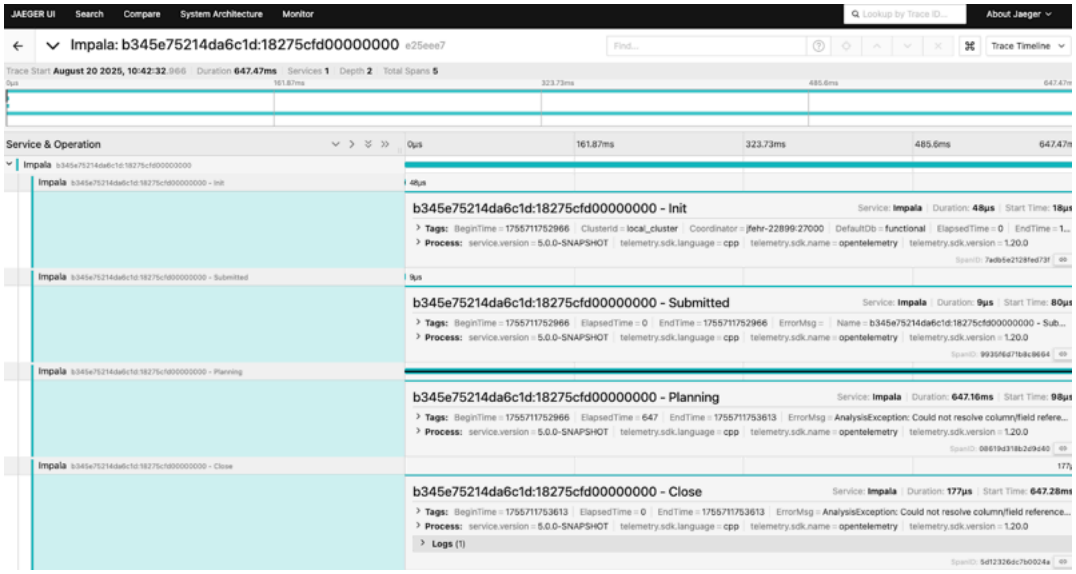


Analyzing child spans

By inspecting the individual child spans, you can gain deep insights into the performance of each stage of the query. For example, you can see how much time was spent on planning, admission control, and execution.

The following query stages are shown in the child spans:

1. Query initialization – This span is generated immediately when a query is received to capture the initial setup and is useful for checking early details, such as the user and query ID.
2. Query planning – This span shows the time taken to plan the query. A long duration here might indicate a complex query or schema-related issues.
3. Admission control – This span records the time spent waiting for resources. A long duration here suggests high resource contention.
4. Query execution – This span represents the time for the actual query to run. This is a crucial metric for evaluating performance.
5. Query close – This span marks the end of the query lifecycle, including cleanup and final reporting.
6. Failed query – A failed query will also generate a trace that can be analyzed to understand the cause of the failure. In this example, the query failed during the planning phase, likely due to a syntax error or a non-existent table. The root span is marked with an error tag, and the logs provide details on the failure.



Analyzing these spans helps you identify performance bottlenecks and understand the entire lifecycle of a query within your Impala Virtual Warehouse.

Limitation of OpenTelemetry support for Impala

Learn about the current limitations of OpenTelemetry support in Impala, including restricted scope and lack of customization.

The current OpenTelemetry integration for Impala has the following limitations:

Telemetry data scope

The scope of telemetry data is currently limited to only traces of select queries. Other types of data such as metrics or logs will not be handled in the initial release. However, the design is prepared to allow metrics or logs to be added in the future without significant architectural changes.

Select queries that leverage Common Table Expressions will not have traces generated for them.

Fixed trace data

The traces being sent from Impala are not configurable. This means you cannot customize the specific data points included in the traces.

Troubleshooting issues in Cloudera Data Warehouse on premises

Get help and resources for troubleshooting issues in Cloudera Data Warehouse on premises.

Note: The timestamp in the filenames of the files inside a diagnostic bundle is as per the time zone configured in Cloudera Manager. However, if you directly list the files in HDFS, the timestamp in the filenames is as per UTC, but the file contents are as per the time zone configured in Cloudera Manager.

Locating Cloudera Data Warehouse on premises logs

Learn how you can access logs for Cloudera Data Warehouse on premises.

About this task

When you generate logs using the Collect Diagnostic Bundles option from the environment, they are written to a partition on the Hive sys.logs table and are stored in the following location on HDFS:

```
/warehouse/[***ENVIRONMENT-NAMEabcde***]/[***DATABASE-CATALOG***]/warehouse/
tablespace/external/hive/sys.db/logs
```

“abcde” is a random 5-character string that is appended to the environment name.

These partitions are retained for 7 days by default.

Procedure

1. Log in to the OpenShift or Experiences Compute Service (ECS) cluster and determine the location of the sys.logs table by running the following query:

   ```
   DESCRIBE FORMATTED sys.logs;
   ```

   This SQL statement returns information about the location of the table which contains the logs.
2. Use the location obtained in Step 1 to locate the Cloudera Data Warehouse on premises logs on the OpenShift or ECS clusters.

Downloading diagnostic bundles in Cloudera Data Warehouse on premises

You can download diagnostic bundles for troubleshooting a Virtual Warehouse in Cloudera Data Warehouse on premises. The diagnostic bundles contain log files for the sidecar containers that support Hive, Impala, or Trino components. These diagnostic bundles are stored on HDFS in the form of ZIP files.

About this task

The log files are generated when you run some workloads on your Virtual Warehouse.

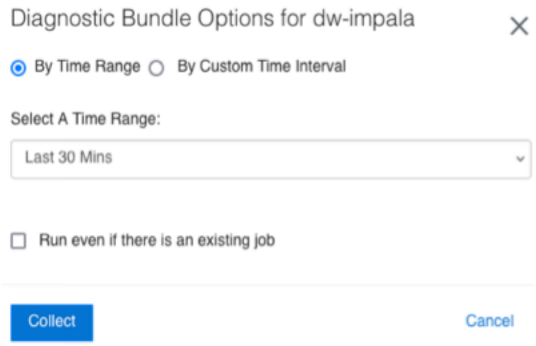
Note: The timestamp in the filenames of the files inside a diagnostic bundle is as per the time zone configured in Cloudera Manager. However, if you directly list the files in HDFS, the timestamp in the filenames is as per UTC, but the file contents are as per the time zone configured in Cloudera Manager.

Before you begin

- Before you can download log files, you must, of course, run workloads on your Hive, Impala, or Trino Virtual Warehouse to generate the logs.

Procedure

1. Log in to the Cloudera Data Warehouse service as a PowerUser.
2. Go to the Virtual Warehouses, locate the Virtual Warehouse from which you want to collect diagnostic data, and click ⋮ Collect Diagnostic Bundle .

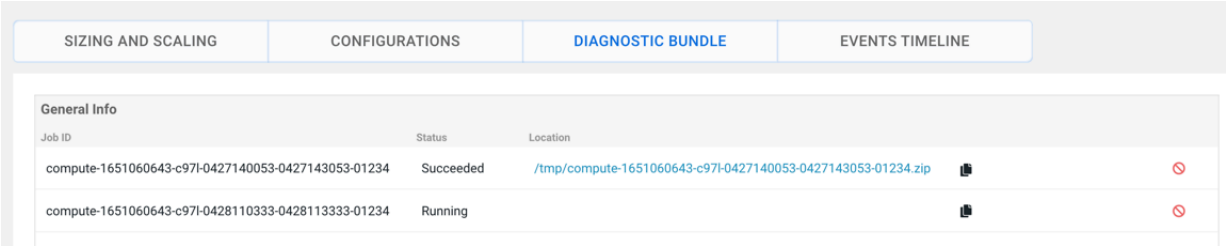


3. Select the time period for which you want to generate the logs.
   - Select the By Time Range option to generate logs from last 30 minutes, one hour, 12 hours, or 24 hours.
   - Select By Custom Time Interval option to generate logs for a specific time period based on your requirement.

   Note: You must set the time range as per the UTC timezone.
4. Select the Run even if there is an existing job option to trigger another diagnostic bundle creation when one job is running.
5. Click Collect.

   The following message is displayed: Collection of Diagnostic Bundle for compute-1651060643-c97l initiated. Please go to details page for more information.
6. Go to the Virtual Warehouses details page by clicking ⋮ Edit .
7. Go to the **DIAGNOSTIC BUNDLE** tab.

   The jobs that have been triggered for generating the diagnostic bundles are displayed, as shown in the following image:



8. Click on the link in the Location column to download the diagnostic bundle to your computer.

   When you extract the diagnostic bundle ZIP file that you downloaded, directories appear for log files and a diag nostic-data-generator.log file, which contains troubleshooting information.

Accessing and generating diagnostic bundles in Cloudera Data Warehouse on premises

Cloudera Data Warehouse collects diagnostic data on workload logs, such as Impala Coordinator, Statefulset, CatalogD logs and stores it in the tmp directory on HDFS. You can download the logs using the Hue File Browser from the base cluster.

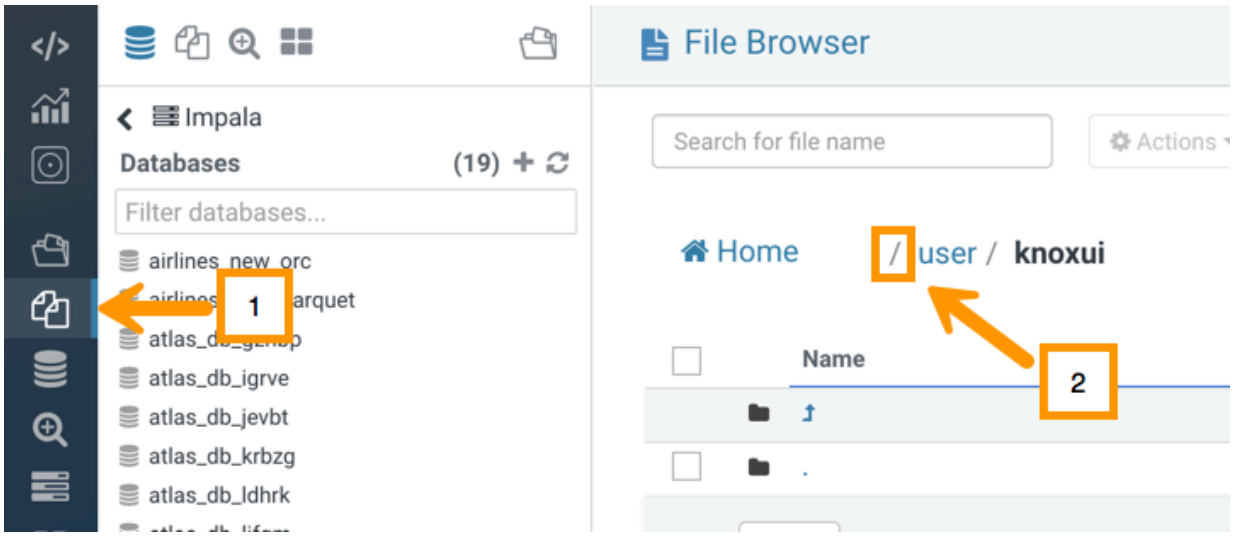
About this task

During the lifetime of a cluster, logs are continously written to the following directory on HDFS: [**WAREHOUSE-DIR**]/warehouse/tablespace/external/hive/sys.db/. When you click Collect Diagnostic Bundle from the Cloudera Data Warehouse web interface, Cloudera Data Warehouse collects the logs for the specified time interval and for the services that you select. These logs are compressed in a ZIP file format and stored in the tmp directory.

Procedure

1. To check the job status and find the HDFS location where the logs are stored, select Edit from the Virtual Warehouse options menu and go to the DIAGNOSTIC BUNDLE tab.

   The logs are collected and bundled under the /tmp/[***VIRTUAL-WAREHOUSE-ID-TIMESTAMP***].zip directory.
2. To access and download the logs, open the Hue service from the base cluster.
3. Go to the Hue File Browser and click the forward slash (/) before the user directory as shown in the following image:



The tmp directory is displayed. You can access and download the logs to your computer by clicking Download.

Impala queries fail

Condition

Impala queries running with high concurrency fail on Cloudera Embedded Container Service (ECS) with the following errors: Invalid or unknown query handle and Invalid session id.

Cause

Impala queries might fail because a single Cloudera Embedded Container Service server may not be able to handle the load. To resolve this issue, enable Cloudera Embedded Container Service High Availability and increase the

Cloudera Embedded Container Service server replicas. This process is called promoting the Cloudera Embedded Container Service agents to servers. You must promote only one Cloudera Embedded Container Service agent at a time. This procedure is explained using an example where you promote the Cloudera Embedded Container Service agent on agent1.example.com and then promote the Cloudera Embedded Container Service agent on agent2.example.com.

Solution

Procedure

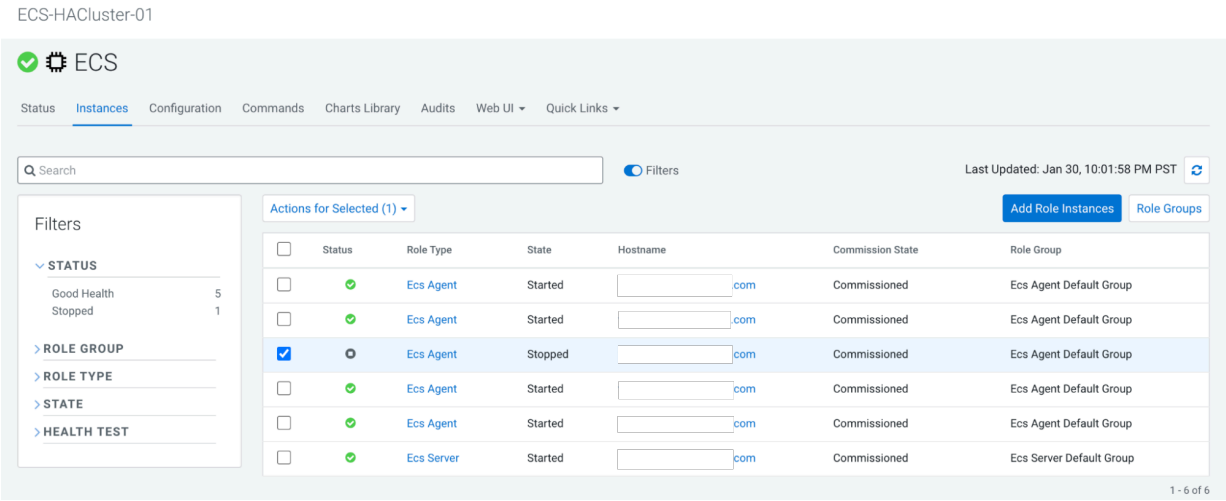
1. Prepare the agent node for promotion by running the following commands on the command line of your Cloudera Embedded Container Service server host.

```
sudo /var/lib/rancher/rke2/bin/kubectl --kubeconfig=/etc/rancher/rke2/rke2.yaml get nodes
```

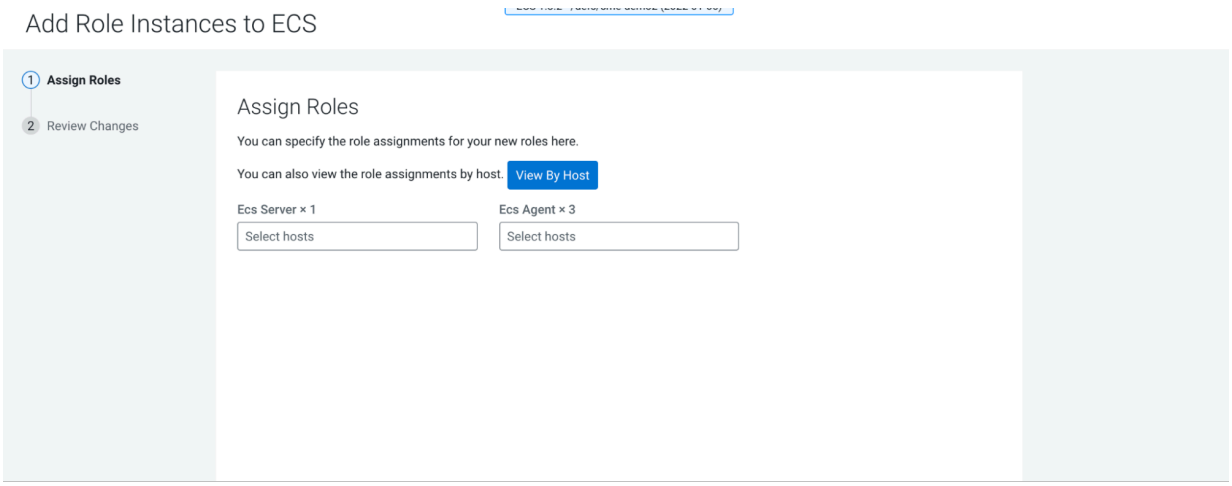
```
sudo /var/lib/rancher/rke2/bin/kubectl --kubeconfig=/etc/rancher/rke2/rke2.yaml drain agent1.example.com --ignore-daemonsets --delete-emptydir-data
```

Note: This may take a few minutes.

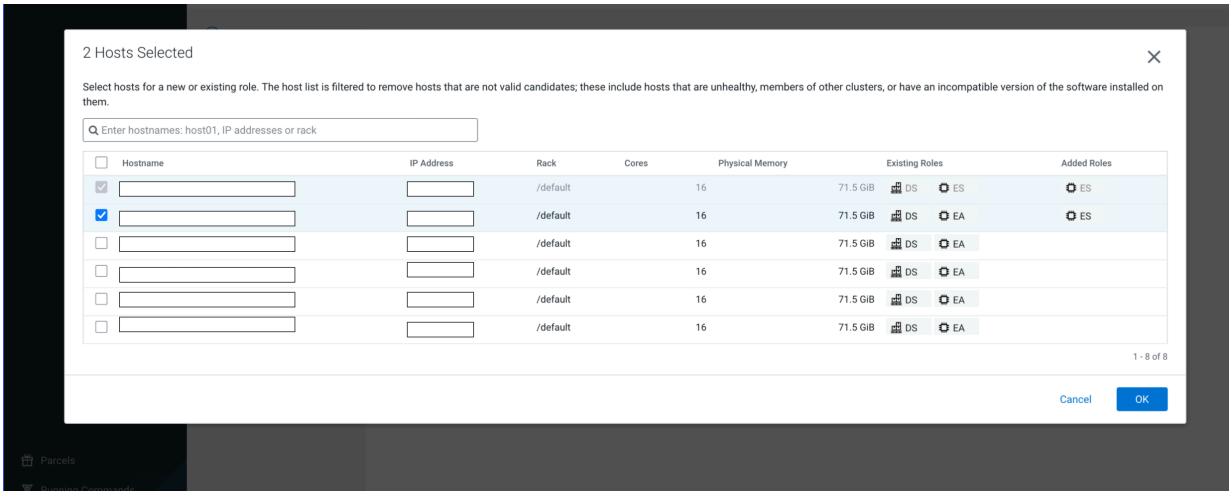
2. In Cloudera Manager, navigate to ECS Cluster ECS . Stop the Cloudera Embedded Container Service Agent running on agent1 and then delete the agent by selecting the respective option from the Actions for Selected drop-down menu.

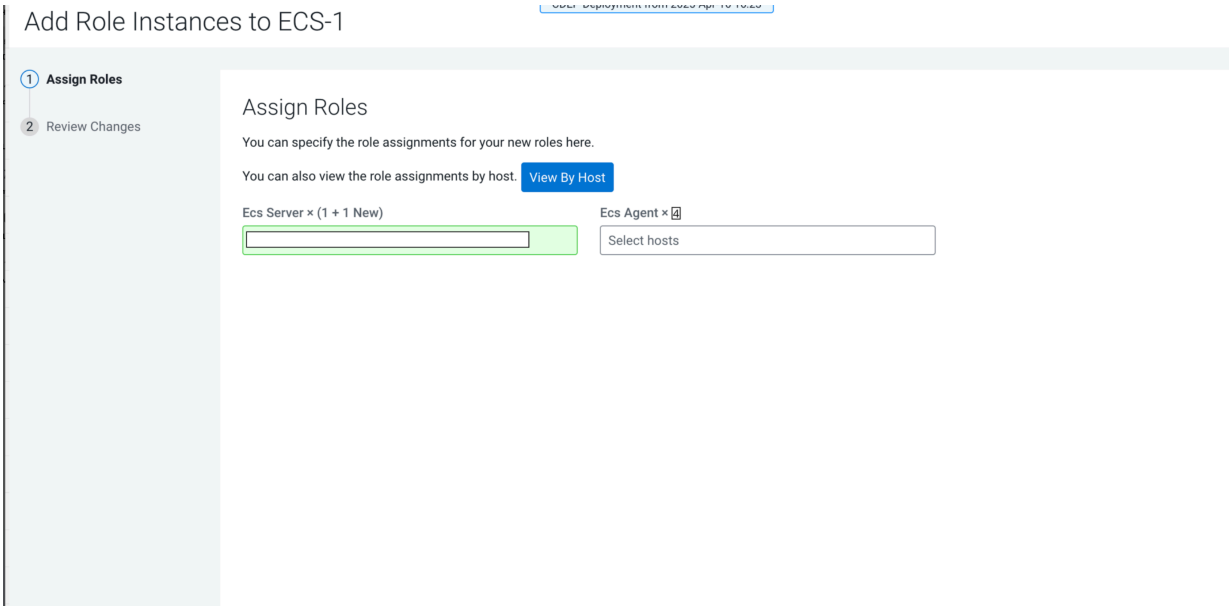


3. In Cloudera Manager, navigate to ECS Cluster ECS and click Add Role Instances.

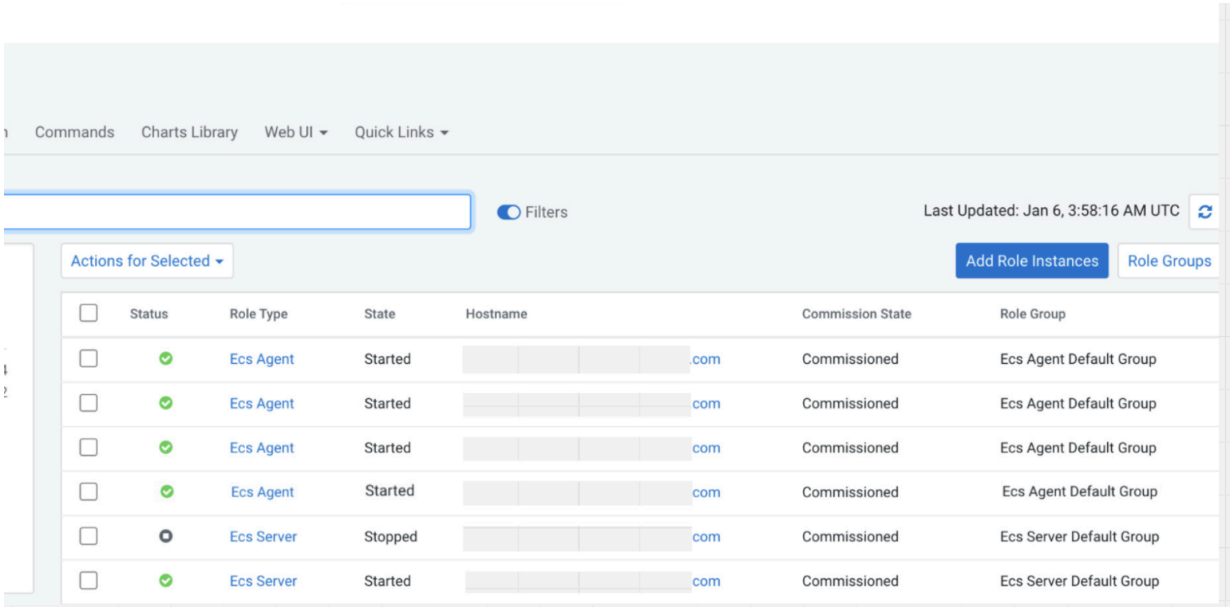


4. Add the available host agent1 as an Cloudera Embedded Container Service server in the Add Role Instances to ECS pop-up. Click Ok.

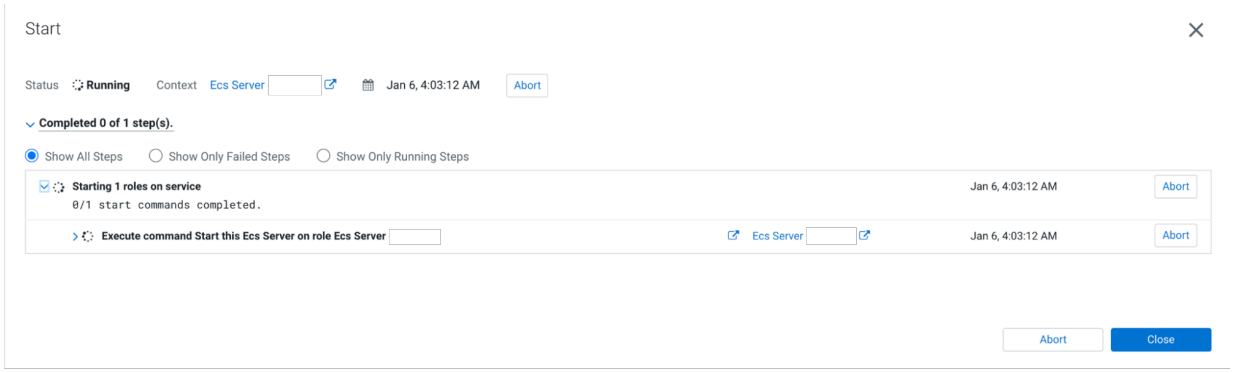




5. Click Continue.



6. Start the new Cloudera Embedded Container Service server from Cloudera Embedded Container Service Instances view. For example, start Cloudera Embedded Container Service Server on agent1.



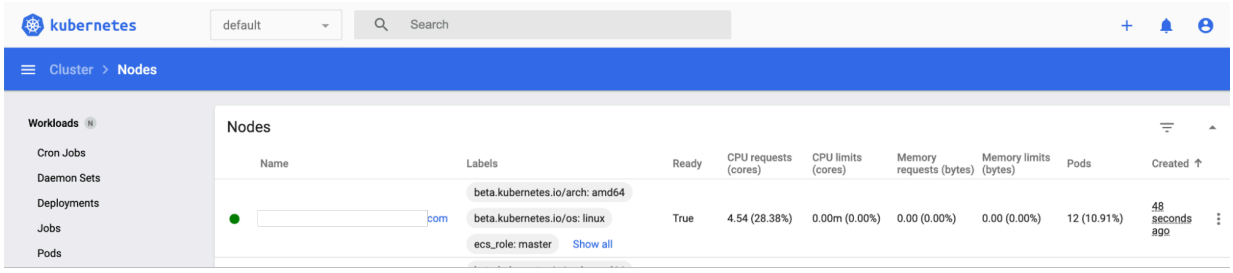
7. On the command line, uncordon the node by running the following command:

```
sudo /var/lib/rancher/rke2/bin/kubectl --kubeconfig=/etc/rancher/rke2/rke2.yaml uncordon agent1.example.com
```

8. Confirm the node's status from webUI or the command line by running the following command:

```
sudo /var/lib/rancher/rke2/bin/kubectl --kubeconfig=/etc/rancher/rke2/rke2.yaml get nodes
```

Note: Do not proceed until node status is Ready. This may take several minutes.



Debugging Impala Virtual Warehouses

You can use the Catalog Web UI, Coordinator Web UI, and the StateStore Web UI to debug Impala Virtual Warehouses in Cloudera Data Warehouse.

Table level events

In addition to global metrics described below, the following table metrics are available for debugging an Impala Virtual Warehouse:

- avg-events-process-duration
- events-consuming-delay-ms

avg-events-process-duration metric

This metric represents the sum of the time for processing all events. This metric is helpful to identify the average duration of processed events on the table and to identify which tables are causing the event-processor to lag behind. As a temporary workaround, you can disable event processing on that table. You can set the metric collection period to 1 minute, 5 minutes, and 15 minutes duration:

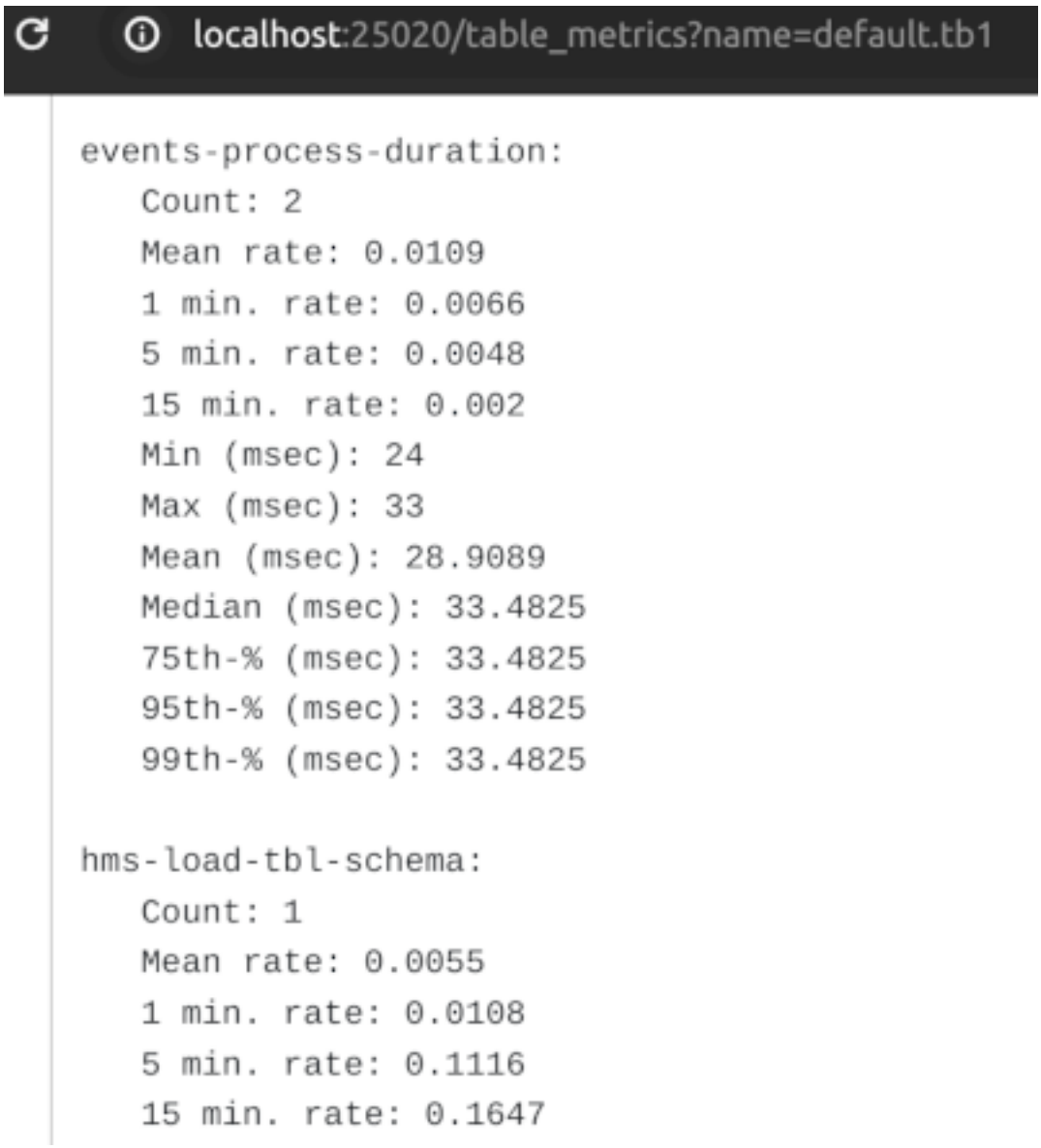
- avg-events-process-duration-1min-rate

  Exponentially weighted moving average (EWMA) of number of events processed in last 1 min
- avg-events-process-duration-5min-rate

  Exponentially weighted moving average (EWMA) of number of events processed in last 5 min
- avg-events-process-duration-1min-rate

  Exponentially weighted moving average (EWMA) of number of events processed in last 15 min

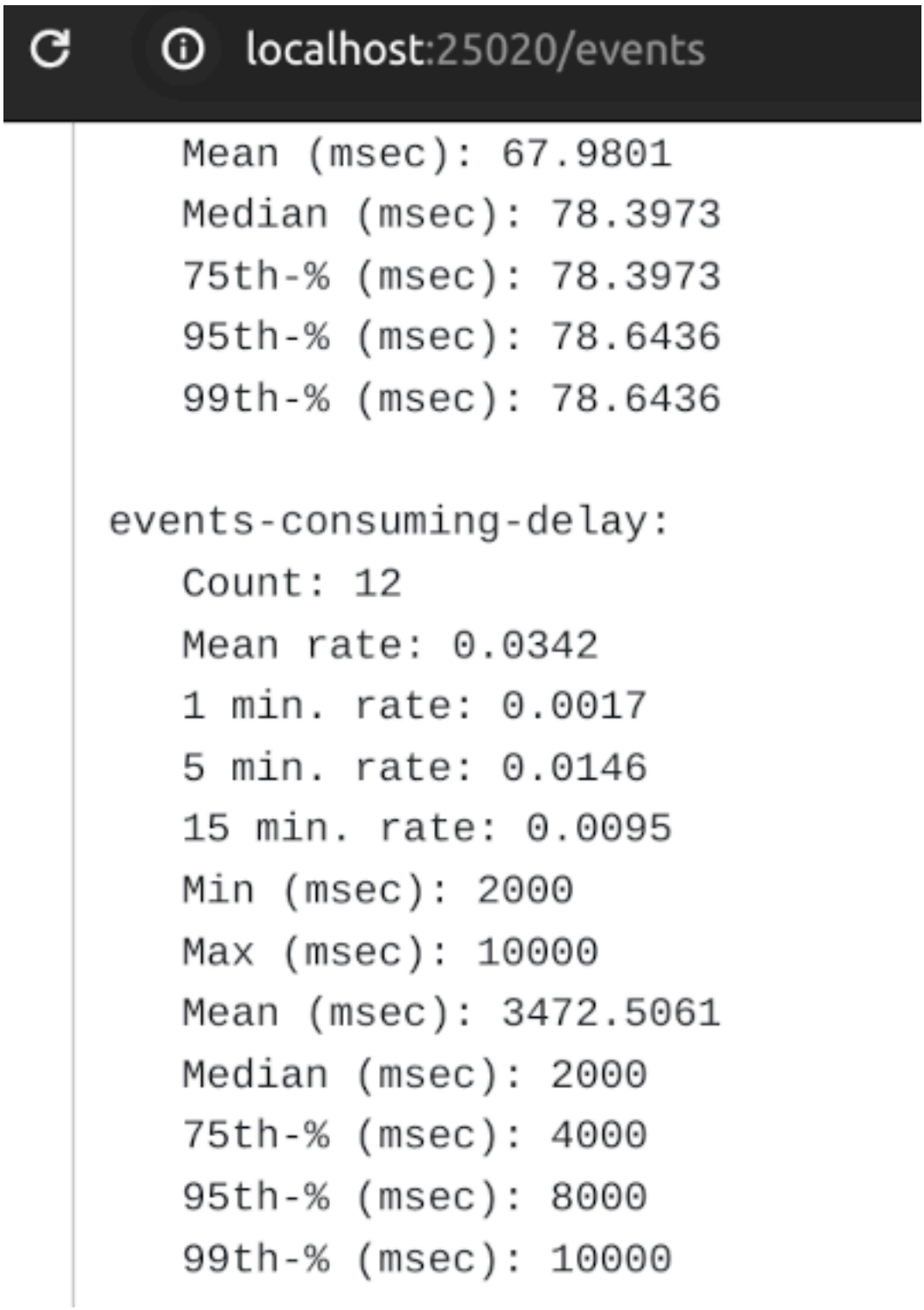
Metric output looks something like this:



events-consuming-delay-ms metric

This metric represents the time difference between creating an event in the metastore and processing an event. Using this metric, you can gauge how long the event processor is lagging.

Metric output looks something like this:



About this task

The Impala daemons (impalad, statestored, and catalogd) debug Web UIs, which can be used in Cloudera Runtime by using Cloudera Manager, is also available in the Cloudera Data Warehouse service. In Cloudera Data Warehouse service, the following Web UIs are provided:

- Impala Catalog Web UI

  This UI provides the same type of information as the Catalog Server Web UI in Cloudera Manager. It includes information about the objects managed by the Impala Virtual Warehouse. For more information about this debug Web UI, see Debug Web UI for Catalog Server.
- Impala Coordinator Web UI

  This UI provides the same type of information as the Impala Daemon Web UI in Cloudera Manager. It includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries. For information about this debug Web UI, see Debug Web UI for Impala Daemon.
- Impala StateStore Web UI

  This UI provides the same type of information as the StateStore Web UI in Cloudera Manager. It includes information about memory usage, configuration settings, and ongoing health checks that are performed by the Impala statestored daemon. For information about this debug Web UI, see Debug Web UI for StateStore.
- Impala Autoscaler Web UI

  This UI gives you insight into autoscaler operations (regular as well as workload-aware autoscaling), accessing log messages, and resetting the log level. The autoscaler Web UI includes information about the queries queued and running, executor groups, suspended calls, scale up/down calls, the autoscaler config, and the autoscaler logs.

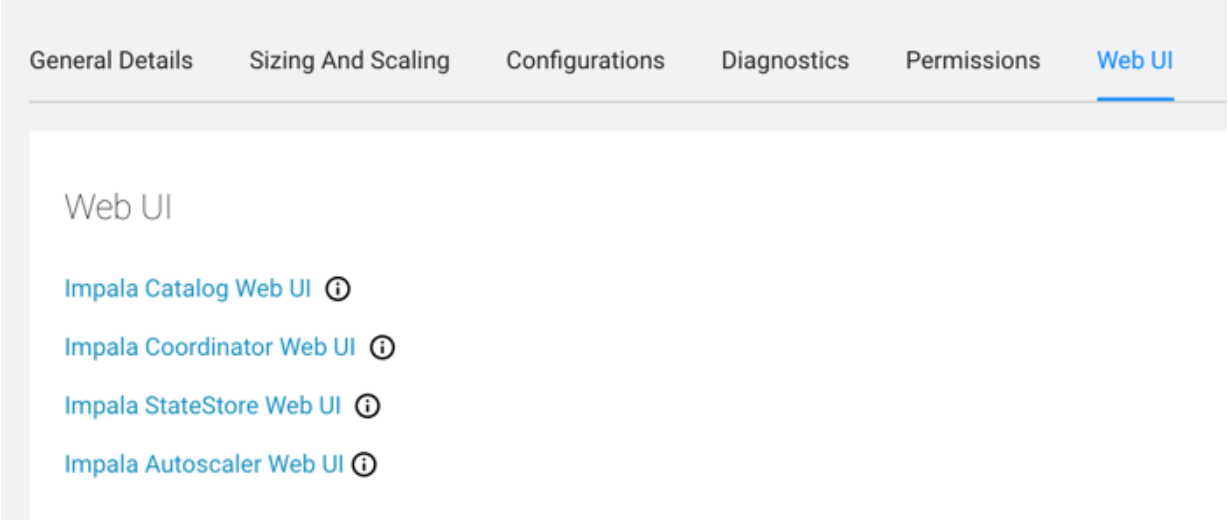
Required role: EnvironmentAdmin

Before you begin

Procedure

1. In the Cloudera Data Warehouse UI on the Overview page, locate the Impala Virtual Warehouse for which you want to view the debug UIs, and select ⋮ Edit .

   The **Virtual Warehouse Details** page is displayed.
2. In the **Virtual Warehouse Details** page, select the Web UI tab. The list of debug Web UI links are displayed as shown in the following image:



3. Click a Web UI link corresponding to an Impala daemon that you want to debug.

   You are prompted to enter your workload user name and password.

Results

After you are authenticated, you can view the debug Web UI and use the information to help you troubleshoot issues with your Impala Virtual Warehouse.

Resolving Kerberos authentication failure

Condition

When you use impala-shell or a JDBC connection with Kerberos as the authentication mechanism, an unauthorized response is received from Impala Virtual Warehouse and the following warning message is visible in the logs of the impala coordinator pod: W0530 12:08:09.118422 21760 authentication.cc:783] Failed to authenticate request from <ip-address>:57978 via SPNEGO: Not authorized: Unspecified GSS failure. Minor code may provide more information: Request ticket server hive/dwx-env-<env-name>.cdp.local@ROOT.HWX.SITE kvno 2 found in keytab but not with enctype des3-hmac-sh.

Cause

DES and DES3 encryption types are deprecated. This error occurs when these deprecated encryption types are in the list of enabled Kerberos encryption types, but the Impala Virtual Warehouse pods are running on a cluster where these insecure encryption types are not supported by the operating system running on the cluster nodes.

Solution

To resolve this problem, you can disable using DES and DES3 encryption types in the Kerberos configuration. On test or development environments, you can include and use deprecated encryption types such as “rc4-hmac”, but you must omit DES and DES3 from the list of the allowed encryption types. Modify the Kerberos configuration of the Cloudera Base on premises cluster by setting appropriate values for the default_tgs_enctypes, default_tkt_enc types, and permitted_enctypes parameters in the libdefaults block of the /etc/krb5.conf file.

For example:

```
default_tgs_enctypes = rc4-hmac aes256-cts aes128-cts
default_tkt_enctypes = rc4-hmac aes256-cts aes128-cts
permitted_enctypes = rc4-hmac aes256-cts aes128-cts
allow_weak_crypto = true
```

Note: The allow_weak_crypto setting is also required to use the deprecated encryption types.

Cloudera recommends that you do not use deprecated encryption types in production environments. Use stronger encryption types such as AES 256 and AES 128. For example:

```
default_tgs_enctypes = aes256-cts aes128-cts
default_tkt_enctypes = aes256-cts aes128-cts
permitted_enctypes = aes256-cts aes128-cts
```

After modifying the Kerberos configuration of the Cloudera Base on premises, go to the Cloudera Data Warehouse web interface and refresh the Dtabase Catalog and the Virtual Warehouse by clicking ⋮ Refresh . This copies configurations from the base cluster to Cloudera Data Warehouse.

Deactivating environments

The Force Delete option offers a convenient way to handle environments stuck in a deleting state, enabling you to remove them directly from the Cloudera Data Warehouse user interface.

Before you begin

The Force Delete option does not perform a clean shutdown. Before using this option, you must delete all cluster components, such as Virtual Warehouses, Data Visualizations, Database Catalogs, and more. Additionally, the Force

Delete option leaves resources on the Kubernetes (k8s) cluster and does not remove the default Database Catalog or log router components. These namespaces must be deleted manually.

Procedure

1. Log in to the Cloudera Data Warehouse service as DWAdmin.
2. Navigate to the Environments tab.
3. Locate the environment you want to deactivate and click Deactivate button.
   The Action dialog box is displayed.
4. In the Action dialog box, select Force Delete option.
5. Click OK.

Cloudera Data Warehouse fails to start after NameNode migration

When you perform an HDFS NameNode migration in Cloudera Base on premises, you may encounter issues with your Cloudera Data Warehouse instance for which you are required to deactivate and reactivate the Cloudera Base on premises environment, and then recreate your Hive and Impala Virtual Warehouses. Learn how you can mitigate these issues without having to deactivate your environment.

Important: If you have enabled quota management in Cloudera Data Warehouse or enabled Active-Active HA in your Impala Virtual Warehouse, it is important that you perform the steps listed below before performing the workaround steps described in the following topics to address the Cloudera Data Warehouse issues resulting due to the NameNode migration.

Enabled quota management

If you have enabled quota management in Cloudera Data Warehouse, you may encounter issues with Hue Pod scheduling during creating, refreshing, rebuilding, or modifying the configuration of Hive or Impala Virtual Warehouses. Perform the following steps to address these issues:

1. Click on the Resource Templates menu on the left navigation pane.
2. Select the Default Resources under the **HIVE** and **IMPALA** tab and then click Actions Make a Copy .
3. Provide a name of the resource template and add resources (increase CPUs and Memory) for any of the components, such as Query Coordinator or Impala Autoscaler, and then click Apply Changes.

   Tip: Select the Set As Default option to make this resource template the default template for all new Cloudera Data Warehouse entities; in this case all new Hive or Impala Virtual Warehouses.
4. Refresh the page on your browser to view the newly created resource template. You can now create your Virtual Warehousing using this newly created resource template.
5. Once the statefulset/deployment of the modified component is present, reset its resources back to its original values. By doing this, you are freeing up enough resources for the Hue front end to get scheduled.

Enabled Active-Active High Availability (HA)

If you have enabled Active-Active HA for your Impala Virtual Warehouse, then you may notice pod start failures in Catalogd pods. To address this issue, you must uncheck the Enable Impala Catalog Server HA option from the **Sizing And Scaling** tab of the **Impala Virtual Warehouse details** page.

Prerequisites for fixing issues after NameNode migration

Learn about the prerequisites that you must perform before fixing issues related to your Database Catalog and Virtual Warehouses after migrating your HDFS NameNode.

Before you begin

Ensure that you meet the following prerequisites before addressing the issues related to Cloudera Data Warehouse:

1. Login to Cloudera Manager and click Clusters HIVE_ON_TEZ .
2. From the **HIVE_ON_TEZ** service page, click Actions Download Client Configuration .

   Save and extract the contents of the hive_on_tez-1-clientconfig.zip file.
3. Open the hdfs-site.xml configuration file and identify the configuration (dfs.ha.namenodes.<NameNode name>) containing the names of the NameNodes. Make a note of the value.

```
 <property>
    <name>dfs.ha.namenodes.ns1</name>
    <value>namenode1546339554,namenode1546335674</value>
  </property>
```

4. Identify and note down all the NameNode specific configuration from the hdfs-site.xml configuration file.

```
<property>
    <name>dfs.ha.namenodes.ns1</name>
    <value>namenode1546339554,namenode1546335674</value>
  </property>
  <property>
    <name>dfs.namenode.rpc-address.ns1.namenode1546339554</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.servicerpc-address.ns1.namenode1546339554</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.http-address.ns1.namenode1546339554</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.https-address.ns1.namenode1546339554</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.rpc-address.ns1.namenode1546335674</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.servicerpc-address.ns1.namenode1546335674</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.http-address.ns1.namenode1546335674</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
  <property>
    <name>dfs.namenode.https-address.ns1.namenode1546335674</name>
    <value>[***HOSTNAME***]:[***PORT***]</value>
  </property>
```

What to do next

Perform the steps described in the following topics to address issues specific to your Database Catalog and Hive or Impala Virtual Warehouses.

Fixing issues with Database Catalog

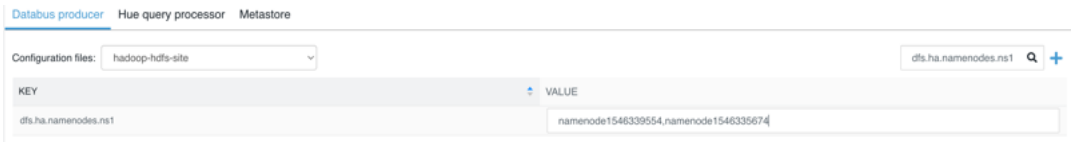
After an HDFS NameNode migration on a Cloudera Base on premises instance, the Database Catalog in Cloudera Data Warehouse can go into an error state. Learn how to address this issue and bring the Database Catalog back to a healthy state.

About this task

Configuration changes, upgrade, or rebuild operations cannot fix this issue because the problem is that the namenode configuration in the Database Catalog remains the same as the one before the namenode migration, which is incorrect and not usable.

Procedure

1. Log in to Cloudera Data Warehouse and click the Database Catalogs tab.
2. Select the required Database Catalog and click ⋮ Edit .
3. From the Database Catalog details page, click CONFIGURATIONS and update the value of dfs.ha.namenodes .ns1 in the hadoop-hdfs-site configuration file for Databus producer, Hue query processor, and Metastore. Use the NameNode value that you saved previously from the prerequisite section.



4. Click Apply Changes and wait for the Database Catalog to reach a healthy state.

Fixing issues with Hive Virtual Warehouse

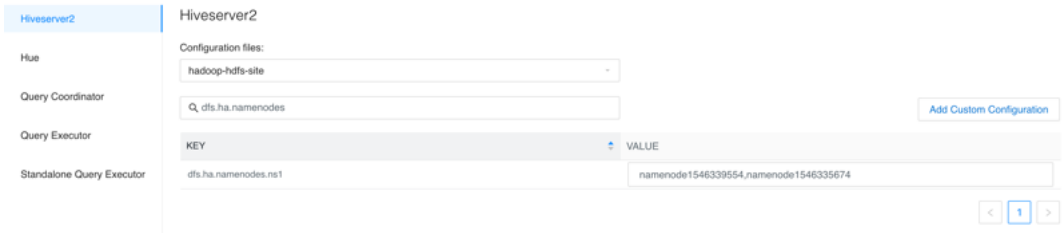
After an HDFS NameNode migration on a Cloudera Base on premises instance, the Hive Virtual Warehouse can go down and become inoperable. Learn how to fix issues with your Hive Virtual Warehouse.

About this task

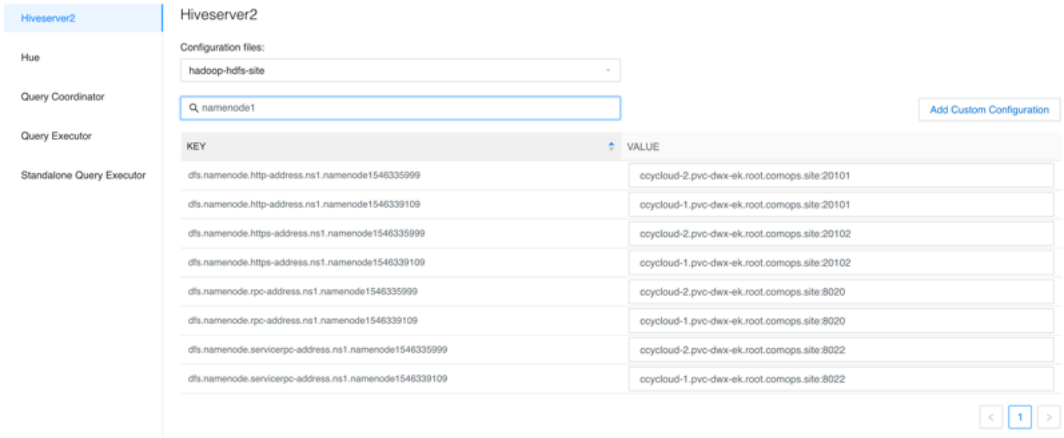
The simplest way to fix this issue is to delete the Hive Virtual Warehouse and recreate it with the same name, configuration, autoscaling parameters, user groups, and so on. However, if deleting the Virtual Warehouse is not an option, then perform the following steps.

Procedure

1. From the Cloudera Data Warehouse **Overview** page, click the Virtual Warehouses tab.
2. Select the required Hive Virtual Warehouse and click ⋮ Edit .
3. From the Virtual Warehouse details page, click Configurations and update the value of dfs.ha.namenodes.ns1 in the hadoop-hdfs-site configuration file for Hiveserver2, Query Coordinator, and Query Executor. Use the NameNode value that you saved previously from the prerequisite section.



4. In the hadoop-hdfs-site configuration file for Hiveserver2, Query Coordinator, and Query Executor, update the remaining NameNode specific configurations that you saved previously from the prerequisites section.



5. Click Apply Changes and wait for the Hive Virtual Warehouse to reach a healthy state.

Fixing issues with Impala Virtual Warehouse

After an HDFS NameNode migration on a Cloudera Base on premises instance, the Hive Virtual Warehouse can go down and become inoperable. Learn how to fix issues with your Impala Virtual Warehouse.

About this task

Like in the case of Database Catalog and Hive Virtual Warehouse, you do not have the option to modify the hadoop-hdfs-site configuraton file in an Impala Virtual Warehouse. Therefore, the only way to fix this issue is to delete the Virtual Warehouse and recreate it with the same name, configuration, autoscaling parameters, user groups, and so on.

Troubleshooting common issues in Impala

This topic describes the general troubleshooting procedures to diagnose some of the commonly encountered issues in Impala.

Symptom	Explanation	Recommendation
Impala takes a long time to start.	Impala instances with large numbers of tables, partitions, or data files take longer to start because the metadata for these objects is broadcast to all `impalad` nodes and cached.	Adjust timeout and synchronicity settings.
Query rejected with the default pool-defined memory limit settings.	Some complex queries fail because the minimum memory reservation per host is greater than the memory available to the query for buffer reservations.	Increase VW t-shirt size so that there are more hosts in the executor group and less memory is needed per host.
Joins fail to complete.	There may be insufficient memory. During a join, data from the second, third, and so on sets to be joined is loaded into memory. If Impala chooses an inefficient join order or join mechanism, the query could exceed the total memory available.	Start by gathering statistics with the COMPUTE STATS statement for each table involved in the join. Consider specifying the [SHUFFLE] hint so that data from the joined tables is split up between nodes rather than broadcast to each node. If tuning at the SQL level is not sufficient, add more memory to your system or join smaller data sets.
Queries return incorrect results.	Impala metadata may be outdated after changes are performed in Hive.	After inserting data, adding a partition, or other operation in Hive, refresh the metadata for the table with the REFRESH statement.

Symptom	Explanation	Recommendation
Attempts to complete Impala tasks such as executing INSERT SELECT statements fail. The Impala logs include notes that files could not be opened due to permission denied.	This can be the result of permissions issues. For example, you could use the Hive shell as the hive user to create a table. After creating this table, you could attempt to complete some action, such as an INSERT SELECT on the table. Because the table was created using one user and the INSERT SELECT is attempted by another, this action may fail due to permissions issues.	Ensure the Impala user has sufficient permissions to the table that the Hive user created.
Impala fails to start up, with the `impalad` logs referring to errors connecting to the statestored service and attempts to re-register.	A large number of databases, tables, partitions, and so on can require metadata synchronization, particularly on startup, that takes longer than the default timeout for the statestore service.	Configure the statestore timeout value and possibly other settings related to the frequency of statestore updates and metadata loading.

Virtual Warehouse Fails to Start

This topic provides steps to troubleshoot issues where the Virtual Warehouse fails to start due to outdated Kerberos settings, resulting in HiveServer2 POD launch failures.

Condition

The Virtual Warehouse (VW) fails to start, and the HiveServer2 POD fails to launch. The following trace is present in the HiveServer2 logs:

```
+ DOWNLOAD_PATH=/aux-jars/
+ '[' true == true ']'
+ SERVICE_KEYTAB=/etc/security/keytabs/hive.service.keytab
+ SERVICE_PRINCIPAL=hive/dwx-env-feddatalakedev-env.cdp.local@US-POCLAB.DEL
LPOC.COM
+ kinit -V -k -t /etc/security/keytabs/hive.service.keytab hive/dwx-env-fed
datalakedev-env.cdp.local@US-POCLAB.DELLPOC.COM
Using default cache: /tmp/krb5cc_1000
Using principal: hive/dwx-env-feddatalakedev-env.cdp.local@US-POCLAB.DELLP
OC.COM
Using keytab: /etc/security/keytabs/hive.service.keytab
kinit: Client's credentials have been revoked while getting initial crede
ntials
+ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1000)
+ [[ -z '' ]]
+ echo 'CDW_HIVE_AUX_JARS_PATH is not defined. Skipping jars download from p
ath..'
+ exit
CDW_HIVE_AUX_JARS_PATH is not defined. Skipping jars download from path..
```

Cause

The issue occurs when Kerberos settings are changed but not updated in Cloudera Data Warehouse, leading to outdated keytab files or credentials cache.

Important: Ensure that you have taken a backup of the environment before performing these steps to avoid losing any configurations.

Solution

Procedure

1. Log in to Cloudera Manager as an administrator.
2. Stop all services in the base cluster, including Management services.
3. Go to Cluster Actions and deploy the Kerberos client configuration.

   Note: If krb5.conf is managed by Cloudera Manager, this step updates it automatically. If krb5.conf is not managed by Cloudera Manager, manually copy the updated /etc/krb5.conf file to all servers in the base cluster.
4. Go to Administration Security Kerberos , select all Kerberos principals, and click Regenerate Keytabs.
5. Start all stopped services.
6. Refresh the environment, database catalog, and all Virtual Warehouses by clicking the refresh option for each respective window. For more information, see *Refresh Cloudera Data Warehouse*.

Alternative Resolution Steps

Procedure

If the above steps do not resolve the issue, perform the following:

1. Delete the Virtual Warehouses.
2. Deactivate the environment.
3. Reactivate the environment.
4. Create the Hive Virtual Warehouses again. See *Add Virtual Warehouse*.

Using Breakpad Minidumps for Crash Reporting

The breakpad project is an open-source framework for crash reporting. Impala can use breakpad to record stack information and register values when any of the Impala-related daemons crash due to an error such as SIGSEGV or unhandled exceptions. The dump files are much smaller than traditional core dump files. The dump mechanism itself uses very little memory, which improves reliability if the crash occurs while the system is low on memory.

Using the Minidump Files for Problem Resolution

You can see in the Impala log files or in the Cloudera Manager charts for Impala when crash events occur that generate minidump files. Because each restart begins a new log file, the “crashed” message is always at or near the bottom of the log file. (There might be another later message if core dumps are also enabled.)

Important: If an Impala-related daemon experiences a crash due to an out-of-memory condition, it does not generate a minidump for that error.

Typically, you provide minidump files to Cloudera Support as part of problem resolution, in the same way that you might provide a core dump. The Send Diagnostic Data under the Support menu in Cloudera Manager guides you through the process of selecting a time period and volume of diagnostic data, then collects the data from all hosts and transmits the relevant information for you.

Procedure

1. In Cloudera Manager, navigate to Impala service Configuration .

2. In the search field, type minidump.
3. Set the following fields to configure breakpad minidumps.
   - minidump_path: Turn on or off generation of the minidump files.

     By default, a minidump file is generated when an Impala-related daemon crashes.
   - minidump_path: Specify the Location for minidump files.

     By default, all minidump files are written to the following location on the host where a crash occurs: /var/log/impala-minidumps/*DAEMON_NAME*

     The minidump files for `impalad`, `catalogd`, and `statestored` are each written to a separate directory.

     If you specify a relative path for this setting, the value is interpreted relative to the default minidump_path directory.
   - max_minidumps: Specify the number of minidump files.

     Like any files used for logging or troubleshooting, consider limiting the number of minidump files, or removing unneeded ones, depending on the amount of free storage space on the hosts in the cluster.

     Because the minidump files are only used for problem resolution, you can remove any such files that are not needed to debug current issues.

     The default for this setting is 9. A zero or negative value is interpreted as “unlimited”.
4. Click Save Changes and restart Impala.
5. To provide minidump files to Cloudera Support as part of problem resolution, in Cloudera Manager, navigate to Support Send Diagnostic Data and follow the steps.

Cloudera CLI for Cloudera Data Warehouse

Cloudera CLI allows you to manage users, environments and other entities in your on premises deployment. After you setup and configure Cloudera CLI on your clusters, you can use the various commands and sub-commands that are available.

Commands for Cloudera Data Warehouse are available here. To see whether a command or a sub-command is available for on premises, check the "Form Factor" section.

Runtime documentation for Cloudera Data Warehouse

Cloudera Data Warehouse Runtime provides tightly integrated Apache Hive, Apache Impala, and Hue services for Cloudera Data Warehouse. In this section, you can read about how to use Apache Hive and Apache Impala SQL from clients that connect to your Virtual Warehouse in Cloudera on premises and how to use Iceberg. Also covered is the Hue interactive query editor, which you can open from a Virtual Warehouse to run SQL queries.

Related Information

Cloudera Data Warehouse Runtime documentation

Runtime component versions for Cloudera Data Warehouse on premises

List of labels for third-party integration

You can integrate third-party apps and services with Cloudera Data Warehouse by using node labels. By tagging and labeling Cloudera Data Warehouse entities and providing the Helm charts of the services you want to integrate, the third-party applications can write Kubernetes injectors to add custom services to Cloudera Data Warehouse

without modifying the Cloudera Data Warehouse Docker images. These services then run as sidecar containers to the Cloudera Data Warehouse pods.

Pod labels for labeling Cloudera Data Warehouse entities

The following table provides a list of labels with accepted values and examples:

Label key	Description	Accepted values
cdw.cloudera.com/application	The application name	• database-catalog • impala • hive • hue
cdw.cloudera.com/component	The component name	• metastore • coordinator • executor • admission-controller • catalog • statestore • hiveserver2 • frontend • backend
cdw.cloudera.com/app-version	The Cloudera Runtime version for Hive, Impala, and Hue	See the Cloudera Data Warehouse component version information
cdw.cloudera.com/cdw-version	The Cloudera Data Warehouse version	See the Cloudera Data Warehouse component version information

Related Information

Version information for Cloudera Data Warehouse on premises components