Pipeline Manager Configuring Health Checks

On this page

Pipeline managers can now be configured to perform health checks to determine how Aspire is performing.

Configuration

To configure health checks for your system add a <healthChecks> section to the pipeline managers for which you desire health checks. Details on how to configure each type of health check are given below. Multiple health checks can be configured per pipeline manager.

The following is an example configuration:

```
 <component name="MyPipelineManager" subType="pipeline" factoryName="aspire-application">
   <healthChecks>

     <timestamp name="Test Timestamp" redThreshold="4000" yellowThreshold="1000"
                history="5" />

     <initialization name="Test Long Initializer">
        <check componentRef="/pipeline/LongInitializer"/>
        <check componentRef="/pipeline/Concat-Test"/>
     </initialization>

     <jobCount redThreshold="1" />

   </healthChecks>

   <!-- Configure your pipelines here -->

   <!-- Configure your components here -->

 </component>
```

Note that all health checks can have a "user friendly name" attached to them (the @name attribute).

Once health checks are configured for your pipeline managers, they will be automatically accumulated and made available as requested.

Types

There are multiple types of health checks available. The types currently available include:

- **Initialization** - Checks to see if components are fully initialized
- **Timestamp** - Timestamps jobs when they start and when they are complete. Jobs which take too long to complete can be flagged as either "RED" or "YELLOW".
- **Job Count** - Provides status on total jobs started, completed successfully, and completed with an error. Health is based on the count of jobs which failed.
- **Latency** - Computes a moving average of the time it takes to complete a job.

Usage

Health Checks can be:

- **INITIALIZING** - Either the Aspire System itself is still loading configurations, or a component has a long initialization which is still in progress.
- **GREEN** - Everything okay
- **YELLOW** - Some health check is showing poor behavior
- **RED** - Jobs are failing or jobs are too slow to satisfy the service levels

The health checks for all pipeline managers across the system are accumulated into a single health check for the entire server. URLS are also available for accessing and managing health checks:

- Entire server health: http://localhost:50505/aspire?cmd=health
- Health check details: http://localhost:50505/aspire?cmd=healthDetail
    - Gives the details for all health checks across all pipeline managers.
- Clearing an individual health check: http://localhost:50505/aspire?cmd=healthClear&id=<health-check-id>
- Clearing all Health Checks: http://localhost:50505/aspire?cmd=healthClearAll .

Health Check: Initialization

This health check is used to check components to see if they are initializing. If they are, the health of the system will be returned as "INITIALIZING".

Configuration

```
<initialization name="Test Long Initializer">
   <check componentRef="/pipeline/LongInitializer"/>
   <check componentRef="/pipeline/Concat-Test"/>
</initialization>
```

- *<check>* - Specifies the component to check.

Attribute

- *@componentRef* - The path name to the component. This *must be* the full path name of the component.

Health details

- This health check returns YELLOW if a component is not available.
- This health check returns RED if there is an error accessing the component or getting the component's status.
    - A health of RED supercedes a health of INITIALIZING.

***Note**: It is expected that the initialization health check will be performed automatically in a future release of Aspire, at which point this health check will be deprecated.*

Health Check: Job Count

This health check provides a total count of jobs and will determine the health of the system based on the total number of failed jobs that occur.

Configuration

```
<jobCount name="Count of Document Jobs" redThreshold="3" yellowThreshold="1"/>
```

Attributes

- *@redThreshold* - If total number of failed jobs is greater than or equal to this number, system health is RED. Should be 1 or more.
    - If @redThreshold is "0", your system will be RED all the time! So, set it to 1 or more.
- *@yellowThreshold* - If total number of failed jobs is greater than or equal to this number, system health is YELLOW. Should be 1 or more.

Health details will show:

- Total count of jobs initiated
- Total count of jobs which completed successfully
- Total count of jobs which failed with an unhandled exception
- Total count of jobs outstanding (equals total initiated minus total completed)

Health Check: Time Stamps

Timestamp health checks are used to check the duration of every job and are typically used for occasional jobs (for example, nightly) that take a long time to run (i.e., hours).

Configuration

```
<timestamp name="Rebuild Dictionary Token Stats" history="5" redThreshold="10000" yellowThreshold="2000"/>
```

Attribute

- ***@history* - The number of past timestamps to display on the health detail page.**
- *@redThreshold* - (milliseconds) If a job takes this much time to complete (or more), health will be RED.
- *@yellowThreshold* - (milliseconds) If a job takes this much time to complete (or more), health will be at least YELLOW.

History

A history of old timestamps will be kept and displayed in the health check details.

Note *that only the most recent timestamp will contribute to the health of the overall system.*

Health Check: Latency

Latency health checks compute a moving average of the time it takes to complete a job and then will flag RED or YELLOW if average job latency rises above specified thresholds.

Averages are computed over a specified number of jobs (@jobsToAverage). This method will also compute a peak average latency and give a history of averages for previous time periods.

Configuration: (defaults to 15 minute intervals over 24 hours)

```
<latency name="Process Single Document" jobsToAverage="5" isSticky="true"
         redThreshold="15000" yellowThreshold="5000" />
```

Configuration: (specify the interval and history length)

```
<latency name="Process Single Document" jobsToAverage="5" isSticky="true"
         redThreshold="15000" yellowThreshold="5000"
         interval="3600000" history="48" />
```

Attributes for Moving Averages

- *@jobsToAverage* - The number of jobs to include in the moving average which is used to compute health.
- *@isSticky* - Specifies whether peak values are "sticky", that is, they hang around until cleared. For example, if your jobs slow down and the health is RED, it will remain RED (i.e., "sticky") even if they start to speed up again. Can be "true" or "false".
- *@redThreshold* - (ms) If the moving average of job latencies rises above this threshold, health will be RED.
- *@yellowThreshold* - (ms) The average latency threshold for declaring the health to be YELLOW.

Attributes for History Presentation

- *@interval* - (ms) The size of each interval in the history. Measured in milliseconds. If not supplied, defaults to 900000 (15 minutes).
- *@history* - (count) The number of history intervals to save. Histories are a rolling window from the current time back through this number of intervals. Defaults to 96 intervals (equals 24 hour's worth of intervals).

Notes

- Histories are independent of moving average computations. The moving averages are computed independently from the history display. Therefore, it is possible for history intervals to show long latencies and for the health status to still be GREEN. This would occur when a history interval contains fewer than the number of jobs specified in @jobsToAverage.
- Don't make @jobsToAverage too small. If this number is too small, then the computations will become unreliable.
- Failed jobs are not added to latency numbers. Latencies are only computed for successful jobs, since these are the only ones which will provide reliable measurement data.