Enable Spark Job Overrides

Contents:

By default, the Trifacta® Self-Managed Enterprise Edition applies configuration to Spark at the global level. All jobs submitted to the connected instance of Spark utilize the same set of Spark properties and settings. As needed, the set of properties can be modified by administrators through the Admin console.

Optionally, flow owners can configure overrides to the default Spark properties at the output object level.

- Overrides are defined as part of the output object. When you configure these values, they are applied each time that the output object is used to generate a set of results.
- These overrides can be applied to ad-hoc or scheduled jobs.

User-specific Spark overrides: If you have enabled user-specific overrides for Spark jobs, those settings take precedence over the settings that are applied through this feature. For more information, see *Configure User-Specific Props for Cluster Jobs*.

Limitations

> **This feature allows administrators to enable the passthrough of properties to Spark, and users can submit any value of an enabled property. Please be careful in choosing the properties that you enable for users to override.**

Property validation:

- You can enable properties that are destructive.
- Property names of whitelisted properties:
  - No validation of property names is provided by the Trifacta application.
  - No check is performed on property names. Invalid property names are ignored.
- Property values:
  - Property values are not compared against the operating environment. For example, a user can input `1000` for the number of `spark.executor.cores`, which causes job failures.
  - Property values are validated against expected type.

Default Overrides

When this feature is enabled, the following properties are available for users to override at job execution time with their preferred values.

> **NOTE:** These properties are always available for override when the feature is enabled.

Spark parameter	Description
spark.driver.memory	Amount of RAM in GB on each Spark node that is made available for the Spark drivers.
spark.executor.memory	Amount of RAM in GB on each Spark node that is made available for the Spark executors.
spark.executor.cores	Number of cores on each Spark executor that is made available to Spark.
transformer.dataframe.checkpoint.threshold	When checkpointing is enabled, the Spark DAG is checkpointed when the approximate number of expressions in this parameter has been added to the DAG. Checkpointing assists in managing the volume of work that is processed through Spark at one time; by checkpointing after a set of steps, the Trifacta platform can reduce the chances of execution errors for your jobs. By raising this number: • You increase the upper limit of steps between checkpoints. • You may reduce processing time. • It may result in a higher number of job failures.

Spark jobs on Azure Databricks:

For Spark jobs executed on Azure Databricks, only the following default override parameters are supported:

Spark parameter	Description
transformer.dataframe.checkpoint.threshold	See above.

During Spark job execution on Azure Databricks:

> **Whenever overrides are applied to an Azure Databricks cluster, the overrides must be applied at the time of cluster creation. As a result, a new Azure Databricks cluster is spun up for the job execution, which may cause the following:**
>
> - **Delay in job execution as the cluster is spun up.**
> - **Increased usage and costs**
> - **After a new Azure Databricks cluster has been created using updated Spark properties for the job, any existing clusters complete executing any in-progress jobs and gracefully terminate based on the idle timeout setting for Azure Databricks clusters.**

- You cannot apply overrides to the Spark dynamic allocation settings on Azure Databricks, including the following parameters. At job execution time, these parameters are discarded, even if they have been added to the whitelisted properties:
  - spark.dynamicAllocation.enabled
  - spark.shuffle.service.enabled
  - spark.executor.instances

For more details on setting these parameters, see *Tune Cluster Performance*.

Enable

Workspace administrators can enable Spark job overrides.

Steps:

1. Login to the application as a workspace administrator.
2. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
3. Locate the following parameter:

```
Enable Custom Spark Options Feature
```

4. Set this parameter to `Enabled`.

Configure Available Parameters to Override

After enabling the feature, workspace administrators can define the Spark properties that are available for override.

Steps:

1. Login to the application as a workspace administrator.
2. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
3. Locate the following parameter:

```
Spark Whitelist Properties
```

4. Enter a comma-separated list of Spark properties. For example, the entry for adding the following two properties looks like the following:

```
spark.driver.extraJavaOptions,spark.executor.extraJavaOptions
```

5. Save your changes.
6. After saving, please reload the page for the feature to be enabled.
7. When users configure ad-hoc or scheduled Spark jobs, the above properties are now available for overriding, in addition to the default override properties.

Apply Overrides

Overrides are applied to output objects associated with a flow.

Run Job page

When you configure an on-demand job to run:

1. Select the output object in the flow. Then, click **Run**.
2. In the Run Job page, select **Spark** for the Running Environment.
3. Click the Advanced environment options caret.
4. The Spark Execution Properties are available for override.

> **NOTE:** No validation of the property values is performed against possible values or the connected running environment.

For more information, see *Spark Execution Properties Settings*.

For scheduled jobs

When you are scheduling a job:

1. Select the output object in the flow. In the context panel, select the Destinations tab.
2. Under Scheduled destinations, click **Add** to add a new destination or **Edit** to modify an existing one.
3. In the Scheduled Publishing settings page, select **Spark** for the Running Environment.
4. Click the Advanced environment options caret.
5. The Spark Execution Properties are available for override.

> **NOTE:** No validation of the property values is performed against possible values or the connected running environment.

For more information, see *Spark Execution Properties Settings*.

Via API

You can submit Spark property overrides as part of the request body for an output object. See *API Workflow - Manage Outputs*.