Cluster node initialization scripts

March 06, 2023

An init script is a shell script that runs during startup of each cluster node *before* the Apache Spark driver or worker JVM starts.

Some examples of tasks performed by init scripts include:

- Install packages and libraries not included in Databricks Runtime. To install Python packages, use the Databricks `pip` binary located at `/databricks/python/bin/pip` to ensure that Python packages install into the Databricks Python virtual environment rather than the system Python environment. For example, `/databricks/python/bin/pip install <package-name>`.
- Modify the JVM system classpath in special cases.
- Set system properties and environment variables used by the JVM.
- Modify Spark configuration parameters.

> **Warning!**
>
> Databricks scans the reserved location `/databricks/init` for legacy global init scripts. Databricks recommends you avoid storing init scripts in this location to avoid unexpected behavior.

In this article:

Init script types

Databricks supports two kinds of init scripts: cluster-scoped and global.

- **Cluster-scoped**: run on every cluster configured with the script. This is the recommended way to run an init script.
- **Global**: run on every cluster in the workspace. They can help you to enforce consistent cluster configurations across your workspace. Use them carefully because they can cause unanticipated impacts, like library conflicts. Only admin users can create global init scripts. Global init scripts are not run on model serving clusters.

> **Warning!**
>
> Legacy global init scripts and cluster-named init scripts are *deprecated* and cannot be used in new workspaces starting February 21, 2023:
>
> - **Cluster-named**: run on a cluster with the same name as the script. Cluster-named init scripts are best-effort (silently ignore failures), and attempt to continue the cluster launch process. Use Cluster-scoped init scripts instead, they are a complete replacement.
> - **Legacy global**: run on every cluster. They are less secure than the new global init script framework, silently ignore failures, and cannot reference environment variables. Migrate existing legacy global init scripts to the new global init script framework. See Migrate from legacy to new global init scripts.

Whenever you change any type of init script, you must restart all clusters affected by the script.

Init script execution order

The order of execution of init scripts is:

1. Legacy global (deprecated - do not use)
2. Cluster-named (deprecated - do not use)
3. Global
4. Cluster-scoped

Environment variables

Cluster-scoped and global init scripts support the following environment variables:

- `DB_CLUSTER_ID`: the ID of the cluster on which the script is running. See Clusters API 2.0.
- `DB_CONTAINER_IP`: the private IP address of the container in which Spark runs. The init script is run inside this container. See SparkNode.
- `DB_IS_DRIVER`: whether the script is running on a driver node.
- `DB_DRIVER_IP`: the IP address of the driver node.
- `DB_INSTANCE_TYPE`: the instance type of the host VM.
- `DB_CLUSTER_NAME`: the name of the cluster the script is executing on.
- `DB_IS_JOB_CLUSTER`: whether the cluster was created to run a job. See Create a job.

For example, if you want to run part of a script only on a driver node, you could write a script like:

Bash

```bash
echo $DB_IS_DRIVER
if [[ $DB_IS_DRIVER = "TRUE" ]]; then
  <run this part only on driver>
else
  <run this part only on workers>
fi
<run this part on both driver and workers>
```

You can also configure custom environment variables for a cluster and reference those variables in init scripts.

Use secrets in environment variables

You can use any valid variable name when you reference a secret. Access to secrets referenced in environment variables is determined by the permissions of the user who configured the cluster. Secrets stored in environmental variables are accessible by all users of the cluster, but are redacted from plaintext display in the normal fashion as secrets referenced elsewhere.

For more details, see Reference a secret in an environment variable.

Logging

Init script start and finish events are captured in cluster event logs. Details are captured in cluster logs. Global init script create, edit, and delete events are also captured in account-level audit logs.

Init script events

Cluster event logs capture two init script events: `INIT_SCRIPTS_STARTED` and `INIT_SCRIPTS_FINISHED`, indicating which scripts are scheduled for execution and which have completed successfully. `INIT_SCRIPTS_FINISHED` also captures execution duration.

Global init scripts are indicated in the log event details by the key `"global"` and cluster-scoped init scripts are indicated by the key `"cluster"`.

> **Note**
>
> Cluster event logs do not log init script events for each cluster node; only one node is selected to represent them all.

Init script logs

If cluster log delivery is configured for a cluster, the init script logs are written to `/<cluster-log-path>/<cluster-id>/init_scripts`. Logs for each container in the cluster are written to a subdirectory called `init_scripts/<cluster_id>_<container_ip>`. For example, if `cluster-log-path` is set to `cluster-logs`, the path to the logs for a specific container would be: `dbfs:/cluster-logs/<cluster-id>/init_scripts/<cluster_id>_<container_ip>`.

If the cluster is configured to write logs to DBFS, you can view the logs using the File system utility (dbutils.fs) or the DBFS CLI. For example, if the cluster ID is `1001-234039-abcde739`:

Bash

```bash
dbfs ls dbfs:/cluster-logs/1001-234039-abcde739/init_scripts
```

Console

```console
1001-234039-abcde739_10_97_225_166
1001-234039-abcde739_10_97_231_88
1001-234039-abcde739_10_97_244_199
```

Bash

```bash
dbfs ls dbfs:/cluster-logs/1001-234039-abcde739/init_scripts/1001-234039-abcde739_10_97_225_166
```

Console

```console
<timestamp>_<log-id>_<init-script-name>.sh.stderr.log
<timestamp>_<log-id>_<init-script-name>.sh.stdout.log
```

When cluster log delivery is not configured, logs are written to `/databricks/init_scripts`. You can use standard shell commands in a notebook to list and view the logs:

Bash Copy

```bash
%sh
ls /databricks/init_scripts/
cat /databricks/init_scripts/<timestamp>_<log-id>_<init-script-name>.sh.stdout.log
```

Every time a cluster launches, it writes a log to the init script log folder.

> **Important!**
>
> Any user who creates a cluster and enables cluster log delivery can view the `stderr` and `stdout` output from global init scripts. You should ensure that your global init scripts do not output any sensitive information.

Audit logs

Databricks audit logs capture global init script create, edit, and delete events under the event type `globalInitScripts`. See Configure audit logging.

Cluster-scoped init scripts

Cluster-scoped init scripts are init scripts defined in a cluster configuration. Cluster-scoped init scripts apply to both clusters you create and those created to run jobs.

You can configure cluster-scoped init scripts using the UI, the CLI, and by invoking the Clusters API. This section focuses on performing these tasks using the UI. For the other methods, see Databricks CLI setup & documentation and Clusters API 2.0.

You can add any number of scripts, and the scripts are executed sequentially in the order provided.

If a cluster-scoped init script returns a non-zero exit code, the cluster launch *fails*. You can troubleshoot cluster-scoped init scripts by configuring cluster log delivery and examining the init script log.

Cluster-scoped init script locations

You can put init scripts in a cloud storage directory accessible by a cluster.

Example: Use conda to install Python libraries

With Databricks Runtime 9.0 and above, you cannot use conda to install Python libraries. For instructions on how to install Python packages on a cluster, see Libraries.

> **Important!**
>
> Anaconda Inc. updated their terms of service for anaconda.org channels in September 2020. Based on the new terms of service you may require a commercial license if you rely on Anaconda's packaging and distribution. See Anaconda Commercial Edition FAQ for more information. Your use of any Anaconda channels is governed by their terms of service.
>
> As a result of this change, Databricks has removed the default channel configuration for the Conda package manager. This is a breaking change. You must update the usage of conda commands in init-scripts to specify a channel using `-c`. If you do not specify a channel, conda commands will fail with `PackagesNotFoundError`.

In Databricks Runtime 8.4 ML and below, you use the Conda package manager to install Python packages. To install a Python library at cluster initialization, you can use a script like the following:

Bash

```bash
#!/bin/bash
set -ex
/databricks/python/bin/python -V
. /databricks/conda/etc/profile.d/conda.sh
conda activate /databricks/python
conda install -c conda-forge -y astropy
```

Configure a cluster-scoped init script using the UI

This section containts instructions for configuring a cluster to run an init script using the Databricks UI.

> **Important!**
>
> - The script must exist at the configured location. If the script doesn't exist, the cluster will fail to start or be autoscaled up.
> - The init script cannot be larger than 64KB. If a script exceeds that size, the cluster will fail to launch and a failure message will appear in the cluster log.

To use the UI to configure a cluster to run an init script:

1. On the cluster configuration page, click the **Advanced Options** toggle.
2. At the bottom of the page, click the **Init Scripts** tab.

3. In the **Destination** drop-down, select the S3 destination type.
4. Specify a path to the init script.
5. For destination type S3:
   a. Select a region.
   b. Ensure that the cluster is configured with an instance profile that has the `GetObject` permission for access to the bucket. For example:

JSON　　Copy

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::<my-s3-bucket>/*"
      ]
    }
  ]
}
```

6. Click **Add**.

To remove a script from the cluster configuration, click the ✖ at the right of the script. When you confirm the delete you will be prompted to restart the cluster. Optionally you can delete the script file from the location you uploaded it to.

Global init scripts

A global init script runs on every cluster created in your workspace. Global init scripts are useful when you want to enforce organization-wide library configurations or security screens. Only admins can create global init scripts. You can create them using either the UI or REST API.

Important!

Use global init scripts carefully:

- It is easy to add libraries or make other modifications that cause unanticipated impacts. Whenever possible, use cluster-scoped init scripts instead.

- Any user who creates a cluster and enables cluster log delivery can view the `stderr` and `stdout` output from global init scripts. You should ensure that your global init scripts do not output any sensitive information.

You can troubleshoot global init scripts by configuring cluster log delivery and examining the init script log.

Add a global init script using the UI

To configure global init scripts using the Admin Console:

1. Go to the Admin Console and click the **Global Init Scripts** tab.
2. Click **+ Add**.
3. Name the script and enter it by typing, pasting, or dragging a text file into the **Script** field.

   Note

   The init script cannot be larger than 64KB. If a script exceeds that size, an error message appears when you try to save.

4. If you have more than one global init script configured for your workspace, set the order in which the new script will run.
5. If you want the script to be enabled for all new and restarted clusters after you save, toggle **Enabled**.

   Important!

   When you add a global init script or make changes to the name, run order, or enablement of init scripts, those changes do not take effect until you restart the cluster.

6. Click **Add**.

Add a global init script using Terraform

You can add a global init script by using the Databricks Terraform provider and databricks_global_init_script.

Edit a global init script using the UI

1. Go to the Admin Console and click the **Global Init Scripts** tab.
2. Click a script.

3. Edit the script.

4. Click **Confirm**.

Configure a global init script using the API

Admins can add, delete, re-order, and get information about the global init scripts in your workspace using the Global Init Scripts API 2.0.

Migrate from legacy to new global init scripts

If your Databricks workspace was launched before August 2020, you might still have legacy global init scripts. Legacy global init scripts have been deprecated and are no longer available on new workspaces. They should not be used.

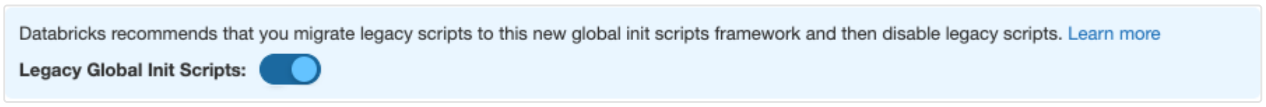
To migrate from legacy global init scripts to the new global init scripts:

1. Copy your existing legacy global init scripts and add them to the new global init script framework using either the UI or the REST API.

   Keep them disabled until you have completed the next step.

2. Disable all legacy global init scripts.

   In the Admin Console, go to the **Global Init Scripts** tab and toggle off the **Legacy Global Init Scripts** switch.

   Databricks recommends that you migrate legacy scripts to this new global init scripts framework and then disable legacy scripts. Learn more

   Legacy Global Init Scripts:

3. Enable your new global init scripts.

   On the **Global Init Scripts** tab, toggle on the **Enabled** switch for each init script you want to enable.

4. Restart all clusters.

   - Legacy scripts will not run on new nodes added during automated scale-up of running clusters. Nor will new global init scripts run on those new nodes. You must restart all clusters to ensure that the new scripts run on them and that no existing clusters attempt to add new nodes with no global scripts running on them at all.

   - Non-idempotent scripts may need to be modified when you migrate to the new global init script framework and disable legacy scripts.