User-Defined Functions

Contents:

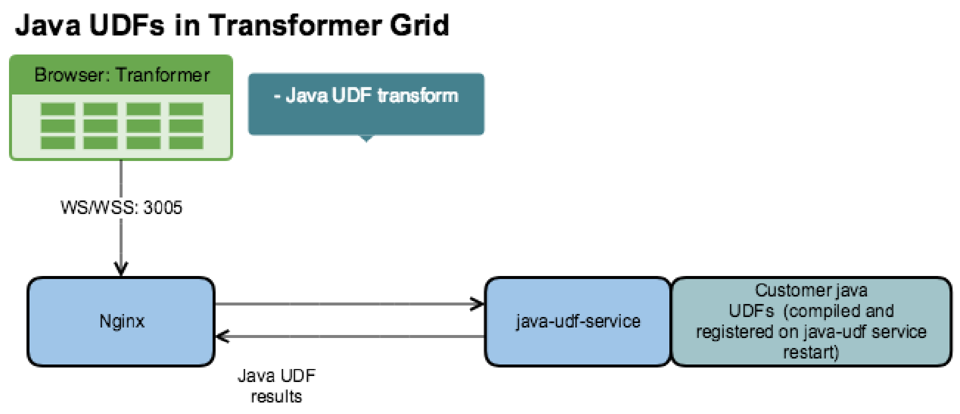
- *UDF Service*
- *Supported UDF Language Frameworks*
- *Running a UDF within the Platform*

The Trifacta® platform enables the creation of user-defined functions (UDFs) for use in your Trifacta deployment. A **user-defined function** is a way to specify a custom process or transformation for use in your specific Trifacta solution, using familiar development languages and third-party libraries. Through UDFs, you can apply enterprise- or industry-specific expertise consistently into your data transformations. A user-defined function is a custom function that is created in one of the supported language frameworks. Each user-defined function has a defined set of inputs and generates a single output.

UDF Service

The following diagram provides a high-level overview of the UDF service which provides integration of user-defined functions into recipe execution.

- Diagram 1: The figure illustrates execution of a UDF in interactive mode, where a user interacts with the Transformer grid.
- Diagram 2: This feature illustrates how UDFs interact with the cluster at job execution time.


Java UDFs in Transformer Grid
Browser: Tranformer
- Java UDF transform
WS/WSS: 3005
Nginx
java-udf-service
Customer java UDFs (compiled and registered on java-udf service restart)
Java UDF results

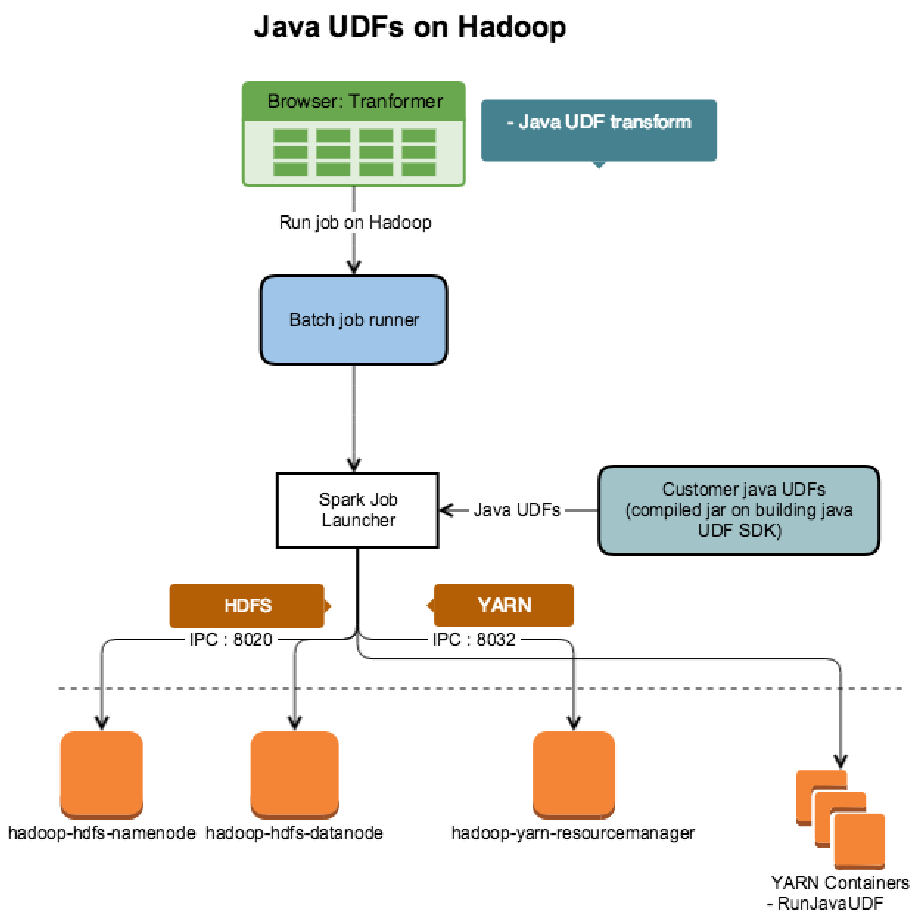


Figure: User-Defined Service

Supported UDF Language Frameworks

Please use the following links to enable the creation of user-defined functions in the listed language.

- *Java UDFs*

Running a UDF within the Platform

After you have created and tested your UDF, you can execute it by entering `udf` in the Search panel and populating the rest of the step in the Transform Builder. In this example, the `AdderUDF` function is executed:

```
udf name:'AdderUDF' col:column1 args:'1' as:'udf_output'
```

Notes:

- The `udf` command causes the named UDF to run.
- After you type `name`, your UDF should appear in a drop-down list. If not, please verify that it has been properly created, compiled, and registered and that the udf-service has been restarted.
- The `col` argument is a comma-separated list of the source data to be used as inputs to the exec method.
- The `args` argument is a string of comma-separated values used as inputs to the init method.
- Optionally, the `as` parameter can be used to provide a specific name to the generated column. If it is not used, a column name is generated.

NOTE: When a recipe containing a user-defined function is applied to text data, any non-printing (control) characters cause records to be truncated by the Spark running environment during job execution. In these cases, please execute the job on the Photon running environment.

NOTE: Running user-defined functions for an external service, such as Hive, is not supported from within a recipe step. As a workaround, you may be able to execute recipes containing such external UDFs on the Photon running environment. Performance issues should be expected on larger datasets.

See *Transformer Page*.