Quick Start Guide

Contents

This page explains how to get started with Framework. Use this page to onboard or evaluate Framework.

Overview

In this quick start guide, you use Framework to transform a `DataRecord` into a `tf.Example` by creating and running a `TransformPlan`. After the transform, you can inspect the generated `tf.Example` sample record with a notebook.

To get started with Framework, do the following:

1. Generate a TransformPlan
2. Transform a DataRecord
3. Read the tf.Example Output

After you complete this quick start guide, use the procedures in this page as a template for your own use case. For example, you could use the generated sample record in a training or validation dataset for a machine learning model.

> **Note**
>
> For instructions for integrating Framework in a machine learning pipeline, see Build Your First Pipeline in the Model Training customer journey.

Prerequisites

To complete the procedures in this page, you must have the following:

- **Access to cloud storage:** For instructions, see Gain Permissions for Cloud Storage.
- **A DataRecord specification JSON file in cloud storage:** For instructions, see Upload to Cloud Storage.

In this quick start guide, we assume you are familiar with the following:

- **DataRecord:** See About DataRecord.
- **Cloud Storage:** See the Cloud Storage documentation.
- **Notebooks:** See the Notebook documentation.

Generate a `TransformPlan`

In this section, you use Framework transform builders to create a `TransformPlan` from `DatasetMetadata`, serialize that plan to JSON, and then upload it to cloud storage.

For a full example of the following code, see the ExampleTransformGenerator source code.

> **Tip**
>
> When you create your own class file, use the ExampleTransformGenerator source code as a template.

To generate a `TransformPlan` from Framework metadata:

1. Load `FeatureContext` in your generator code and use it to build your Framework metadata:

   ```
   FeatureContext fc = ContextUtil.loadFromStorage (
     "PROJECT-ID"
     "STORAGE-NAME"
     "PATH-TO-DATA-SPEC"
     "DATA-SPEC-NAME"
   );

   DatasetMetadata datasetMetadata =
   DatasetMetadataBuilder.buildDatasetMetadata(fc);
   ```

   Where `PATH-TO-DATA-SPEC` is the path to your JSON file in cloud storage.
2. Use the `DatasetMetadata` in your generator code to build and export the `TransformPlan`. For example:

   ```
   REDACTED
   ```

3. Create a `BUILD` file that includes a transform generator target. For an example, see the example_transform target source code.
4. Run the generator and upload the `TransformPlan` to cloud storage:

   ```
   bazel bundle GENERATOR-TARGET
   java -jar GENERATOR-JAR

   csutil cp transform_plan json PATH-TO-CLOUD-STORAGE/transform.json
   ```

   For example:

   ```
   $ bazel bundle /framework/transform_configs:example
   $ java -far dist/example_transform-bundle/example_transform.jar
   $ csutil cp transform_plan.json
   cs://framework_examples/read_job/transform_plan.json
   ```

Transform a `DataRecord`

To transform a `DataRecord` to a `tf.Example` using a `TransformPlan`:

1. Create a config for the job. For example:

   ```
   REDACTED
   ```

   Ensure that you replace the highlighted values in the above example with your own cloud storage details.
   For a full example of the above config, see the example.config source code.
2. Create a job. For example:

   ```
   REDACTED
   ```

   For a full example of the above job code, see the FrameworkReadFromCSExample source code.
   The above example uses jobs to transform a `DataRecord` to a `tf.Example` and then serializes the underlying `tf.Record`. For more examples of jobs, see Create a Job.

3. Create a BUILD file that includes a job target. For an example, see the framework_from_cs target source code.
4. Log in to cloud storage. For instructions, see Log in to Cloud Storage.
5. Run the job:

   ```
   bin/config

   bazel bundle JOB-TARGET

   ./bin/config create --jar JOB-JAR \
   STAGING-STORAGE/REGION/JOB-NAME \
   CONFIG-NAME
   ```

   For example:

   ```
   $ bin/config
   $ bazel bundle framework/jobs:framework-from-cs
   $ ./bin/config create --jar framework-from-cs-bundle/framework-from-
   cs.jar \
     cs-staging/us-central/$USER-framework-read-from-cs-example-scala \
     framework/jobs/example.config
   ```

Read the tf.Example Output

In this section, you use a Framework utility to create a parse_spec and then read the generated tf.Example sample record. For notebook examples that use parse_spec, see the parse_spec generation notebook.

To read the tf.Example output:

1. In a notebook, create a parse_spec and read the generated sample record:

   ```
   cs_path = PATH-TO-TFEXAMPLES
   metadata_path = f"{cs_path}/METADATA.json"
   data_path = f"{cs_path}/DATA-PARTITION.gz"

   parse_spec = create_parse_spec_from_framework_json(metadata_path)

   for raw_recrod in tf.data.TFRecordDataset( [data_path],
   compression_type="GZIP").take(2);
       tf.io.parse_example(raw_record, parse_spec)
   ```

   For example:

   ```
   cs_path = 'cs://foo/bar/tfexamples'
   metadata_path = f"{cs_path}/my_dataset_metadata.json"
   data_path = f"{cs_path}/my_partition-1-of-3.gz"

   parse_spec = create_parse_spec_from_framework_json(metadata_path)

   for raw_recrod in tf.data.TFRecordDataset( [data_path],
   compression_type="GZIP").take(2);
       tf.io.parse_example(raw_record, parse_spec)
   ```

Next Steps

Congratulations! You have used Framework to read from a synthetic DataRecord data set and transform a record into a tf.Example. You have also verified that you can read the output data.

You can use the above procedure as a template to create a transform for your own use case. For information about transform builders that might be suitable for your use case, see About Transform Builders.