Feast Documentation

Feast Authors

Dec 14, 2021

CONTENTS

CHAPTER

ONE

CLIENT

class feast.client.**Client**(*options: Optional[Dict[str, str]] = None*, ***kwargs*)

Feast Client: Used for creating, managing, and retrieving features.

apply(*objects: Union[List[Union[*feast.entity.Entity, feast.feature_table.FeatureTable*]], *feast.entity.Entity, feast.feature_table.FeatureTable*], *project: Optional[str] = None*)

Idempotently registers entities and feature tables with Feast Core. Either a single entity or feature table or a list can be provided.

Parameters **objects** – List of entities and/or feature tables that will be registered

Examples

```
>>> from feast import Client
>>> from feast.entity import Entity
>>> from feast.value_type import ValueType
>>>
>>> feast_client = Client(core_url="localhost:6565")
>>> entity = Entity(
>>>     name="driver_entity",
>>>     description="Driver entity for car rides",
>>>     value_type=ValueType.STRING,
>>>     labels={
>>>         "key": "val"
>>>     }
>>> )
>>> feast_client.apply(entity)
```

apply_entity(*entities: Union[List[*feast.entity.Entity*], *feast.entity.Entity*], *project: Optional[str] = None*)

Deprecated. Please see apply().

apply_feature_table(*feature_tables: Union[List[*feast.feature_table.FeatureTable*], *feast.feature_table.FeatureTable*], *project: Optional[str] = None*)

Deprecated. Please see apply().

archive_project(*project*)

Archives a project. Project will still continue to function for ingestion and retrieval, but will be in a read-only state. It will also not be visible from the Core API for management purposes.

Parameters **project** – Name of project to archive

property core_secure: bool

Retrieve Feast Core client-side SSL/TLS setting

Returns Whether client-side SSL/TLS is enabled

property core_url: str
Retrieve Feast Core URL

Returns Feast Core URL string

create_project(*project: str*)
Creates a Feast project

Parameters **project** – Name of project

delete_feature_table(*name: str*, *project: Optional[str] = None*) → None
Deletes a feature table.

Parameters

- **project** – Feast project that this feature table belongs to
- **name** – Name of feature table

get_entity(*name: str*, *project: Optional[str] = None*) → *feast.entity.Entity*
Retrieves an entity.

Parameters

- **project** – Feast project that this entity belongs to
- **name** – Name of entity

Returns Returns either the specified entity, or raises an exception if none is found

get_feature_table(*name: str*, *project: Optional[str] = None*) → *feast.feature_table.FeatureTable*
Retrieves a feature table.

Parameters

- **project** – Feast project that this feature table belongs to
- **name** – Name of feature table

Returns Returns either the specified feature table, or raises an exception if none is found

get_historical_features(*feature_refs: List[str]*, *entity_source: Union[pandas.core.frame.DataFrame, feast.data_source.FileSource, feast.data_source.BigQuerySource]*, *output_location: Optional[str] = None*) → feast.pyspark.abc.RetrievalJob
Launch a historical feature retrieval job.

Parameters

- **feature_refs** – List of feature references that will be returned for each entity. Each feature reference should have the following format: "feature_table:feature" where "feature_table" & "feature" refer to the feature and feature table names respectively.
- **entity_source** (*Union[pd.DataFrame, FileSource, BigQuerySource]*) – Source for the entity rows. If entity_source is a Panda DataFrame, the dataframe will be staged to become accessible by spark workers. If one of feature tables' source is in BigQuery - entities will be upload to BQ. Otherwise to remote file storage (derived from configured staging location). It is also assumed that the column event_timestamp is present in the dataframe, and is of type datetime without timezone information.

  The user needs to make sure that the source (or staging location, if entity_source is a Panda DataFrame) is accessible from the Spark cluster that will be used for the retrieval job.
- **destination_path** – Specifies the path in a bucket to write the exported feature data files

Returns Returns a retrieval job object that can be used to monitor retrieval progress asynchronously, and can be used to materialize the results.

Examples

```
>>> from feast import Client
>>> from feast.data_format import ParquetFormat
>>> from datetime import datetime
>>> feast_client = Client(core_url="localhost:6565")
>>> feature_refs = ["bookings:bookings_7d", "bookings:booking_14d"]
>>> entity_source = FileSource("event_timestamp", ParquetFormat(), "gs://some-
↪bucket/customer")
>>> feature_retrieval_job = feast_client.get_historical_features(
>>>     feature_refs, entity_source)
>>> output_file_uri = feature_retrieval_job.get_output_file_uri()
    "gs://some-bucket/output/
```

get_historical_features_df(*feature_refs: List[str]*, *entity_source: Union[*feast.data_source.FileSource, feast.data_source.BigQuerySource*]*)

Launch a historical feature retrieval job.

Parameters

- **feature_refs** – List of feature references that will be returned for each entity. Each feature reference should have the following format: "feature_table:feature" where "feature_table" & "feature" refer to the feature and feature table names respectively.
- **entity_source** (*Union[FileSource, BigQuerySource]*) – Source for the entity rows. The user needs to make sure that the source is accessible from the Spark cluster that will be used for the retrieval job.

Returns Returns the historical feature retrieval result in the form of Spark dataframe.

Examples

```
>>> from feast import Client
>>> from feast.data_format import ParquetFormat
>>> from datetime import datetime
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.getOrCreate()
>>> feast_client = Client(core_url="localhost:6565")
>>> feature_refs = ["bookings:bookings_7d", "bookings:booking_14d"]
>>> entity_source = FileSource("event_timestamp", ParquetFormat, "gs://some-
↪bucket/customer")
>>> df = feast_client.get_historical_features(
>>>     feature_refs, entity_source)
```

get_online_features(*feature_refs: List[str]*, *entity_rows: List[Dict[str, Any]]*, *project: Optional[str] = None*) → feast.online_response.OnlineResponse

Retrieves the latest online feature data from Feast Serving. :param feature_refs: List of feature references that will be returned for each entity.

Each feature reference should have the following format: "feature_table:feature" where "feature_table" & "feature" refer to the feature and feature table names respectively. Only the feature name is required.

Parameters

- **entity_rows** – A list of dictionaries where each key-value is an entity-name, entity-value pair.
- **project** – Optionally specify the the project override. If specified, uses given project for retrieval. Overrides the projects specified in Feature References if also are specified.

Returns GetOnlineFeaturesResponse containing the feature data in records. Each EntityRow provided will yield one record, which contains data fields with data value and field status metadata (if included).

Examples

```
>>> from feast import Client
>>>
>>> feast_client = Client(core_url="localhost:6565", serving_url="localhost:6566
↪")
>>> feature_refs = ["sales:daily_transactions"]
>>> entity_rows = [{"customer_id": 0},{"customer_id": 1}]
>>>
>>> online_response = feast_client.get_online_features(
>>>     feature_refs, entity_rows, project="my_project")
>>> online_response_dict = online_response.to_dict()
>>> print(online_response_dict)
{'sales:daily_transactions': [1.1,1.2], 'sales:customer_id': [0,1]}
```

ingest(*feature_table: Union[str, feast.feature_table.FeatureTable], source: Union[pandas.core.frame.DataFrame, str], project: Optional[str] = None, chunk_size: int = 10000, max_workers: int = 1, timeout: int = 120*) → None

Batch load feature data into a FeatureTable.

Parameters

- **feature_table** (*typing.Union[str, feast.feature_table.FeatureTable]*) – FeatureTable object or the string name of the feature table
- **source** (*typing.Union[pd.DataFrame, str]*) – Either a file path or Pandas Dataframe to ingest into Feast Files that are currently supported:
  - parquet
  - csv
  - json
- **project** – Feast project to locate FeatureTable
- **chunk_size** (*int*) – Amount of rows to load and ingest at a time.
- **max_workers** (*int*) – Number of worker processes to use to encode values.
- **timeout** (*int*) – Timeout in seconds to wait for completion.

Examples

```
>>> from feast import Client
>>>
>>> client = Client(core_url="localhost:6565")
>>> ft_df = pd.DataFrame(
>>>         {
>>>            "datetime": [pd.datetime.now()],
>>>            "driver": [1001],
>>>            "rating": [4.3],
>>>         }
>>>     )
>>> client.set_project("project1")
>>>
>>> driver_ft = client.get_feature_table("driver")
>>> client.ingest(driver_ft, ft_df)
```

property job_service_secure: bool
Retrieve Feast Job Service client-side SSL/TLS setting

Returns Whether client-side SSL/TLS is enabled

property job_service_url: str
Retrieve Feast Job Service URL

Returns Feast Job Service URL string

list_entities(*project: Optional[str] = None*, *labels: Dict[str, str] = {}*) → List[*feast.entity.Entity*]
Retrieve a list of entities from Feast Core

Parameters

- **project** – Filter entities based on project name
- **labels** – User-defined labels that these entities are associated with

Returns List of entities

list_feature_tables(*project: Optional[str] = None*, *labels: Dict[str, str] = {}*) → List[*feast.feature_table.FeatureTable*]
Retrieve a list of feature tables from Feast Core

Parameters **project** – Filter feature tables based on project name

Returns List of feature tables

list_features_by_ref(*project: Optional[str] = None*, *entities: List[str] = []*, *labels: Dict[str, str] = {}*) → Dict[*feast.feature.FeatureRef*, *feast.feature.Feature*]
Retrieve a dictionary of feature reference to feature from Feast Core based on filters provided.

Parameters

- **project** – Feast project that these features belongs to
- **entities** – Feast entity that these features are associated with
- **labels** – Feast labels that these features are associated with

Returns features>

Return type Dictionary of <feature references

Examples

```
>>> from feast import Client
>>>
>>> feast_client = Client(core_url="localhost:6565")
>>> features = feast_client.list_features(project="test_project", entities=[
↪"driver_id"], labels={"key1":"val1","key2":"val2"})
>>> print(features)
```

list_projects() → List[str]
List all active Feast projects

Returns List of project names

property project: str
Retrieve currently active project

Returns Project name

property serving_secure: bool
Retrieve Feast Serving client-side SSL/TLS setting

Returns Whether client-side SSL/TLS is enabled

property serving_url: str
Retrieve Feast Serving URL

Returns Feast Serving URL string

set_project(*project: Optional[str] = None*)
Set currently active Feast project

Parameters **project** – Project to set as active. If unset, will reset to the default project.

start_offline_to_online_ingestion(*feature_table:* feast.feature_table.FeatureTable, *start:* *datetime.datetime*, *end:* *datetime.datetime*) → feast.pyspark.abc.SparkJob
Launch Ingestion Job from Batch Source to Online Store for given featureTable

Parameters

- **feature_table** – FeatureTable which will be ingested
- **start** – lower datetime boundary
- **end** – upper datetime boundary

Returns Spark Job Proxy object

version()
Returns version information from Feast Core and Feast Serving

CHAPTER

TWO

DATA SOURCE

class feast.data_source.**BigQueryOptions**(*table_ref: str*)

DataSource BigQuery options used to source features from BigQuery query

classmethod from_proto(*bigquery_options_proto: feast.core.DataSource_pb2.BigQueryOptions*)

Creates a BigQueryOptions from a protobuf representation of a BigQuery option

Parameters **bigquery_options_proto** – A protobuf representation of a DataSource

Returns Returns a BigQueryOptions object based on the bigquery_options protobuf

property table_ref

Returns the table ref of this BQ table

to_proto() → feast.core.DataSource_pb2.BigQueryOptions

Converts an BigQueryOptionsProto object to its protobuf representation.

Returns BigQueryOptionsProto protobuf

class feast.data_source.**BigQuerySource**(*event_timestamp_column: str*, *table_ref: str*, *created_timestamp_column: Optional[str] = ''*, *field_mapping: Optional[Dict[str, str]] = None*, *date_partition_column: Optional[str] = ''*)

property bigquery_options

Returns the bigquery options of this data source

to_proto() → feast.core.DataSource_pb2.DataSource

Converts an DataSourceProto object to its protobuf representation.

class feast.data_source.**DataSource**(*event_timestamp_column: str*, *created_timestamp_column: Optional[str] = ''*, *field_mapping: Optional[Dict[str, str]] = None*, *date_partition_column: Optional[str] = ''*)

DataSource that can be used source features

property created_timestamp_column

Returns the created timestamp column of this data source

property date_partition_column

Returns the date partition column of this data source

property event_timestamp_column

Returns the event timestamp column of this data source

property field_mapping

Returns the field mapping of this data source

static from_proto(*data_source*)
Convert data source config in FeatureTable spec to a DataSource class object.

to_proto() → feast.core.DataSource_pb2.DataSource
Converts an DataSourceProto object to its protobuf representation.

class feast.data_source.**FileOptions**(*file_format: feast.data_format.FileFormat*, *file_url: str*)
DataSource File options used to source features from a file

property file_format
Returns the file format of this file

property file_url
Returns the file url of this file

classmethod from_proto(*file_options_proto: feast.core.DataSource_pb2.FileOptions*)
Creates a FileOptions from a protobuf representation of a file option

Parameters **file_options_proto** – a protobuf representation of a datasource

Returns Returns a FileOptions object based on the file_options protobuf

to_proto() → feast.core.DataSource_pb2.FileOptions
Converts an FileOptionsProto object to its protobuf representation.

Returns FileOptionsProto protobuf

class feast.data_source.**FileSource**(*event_timestamp_column: str*, *file_format: feast.data_format.FileFormat*, *file_url: str*, *created_timestamp_column: Optional[str] = ''*, *field_mapping: Optional[Dict[str, str]] = None*, *date_partition_column: Optional[str] = ''*)

property file_options
Returns the file options of this data source

to_proto() → feast.core.DataSource_pb2.DataSource
Converts an DataSourceProto object to its protobuf representation.

class feast.data_source.**KafkaOptions**(*bootstrap_servers: str*, *message_format: feast.data_format.StreamFormat*, *topic: str*)
DataSource Kafka options used to source features from Kafka messages

property bootstrap_servers
Returns a comma-separated list of Kafka bootstrap servers

classmethod from_proto(*kafka_options_proto: feast.core.DataSource_pb2.KafkaOptions*)
Creates a KafkaOptions from a protobuf representation of a kafka option

Parameters **kafka_options_proto** – A protobuf representation of a DataSource

Returns Returns a BigQueryOptions object based on the kafka_options protobuf

property message_format
Returns the data format that is used to encode the feature data in Kafka messages

to_proto() → feast.core.DataSource_pb2.KafkaOptions
Converts an KafkaOptionsProto object to its protobuf representation.

Returns KafkaOptionsProto protobuf

property topic
Returns the Kafka topic to collect feature data from

class feast.data_source.**KafkaSource**(*event_timestamp_column: str*, *bootstrap_servers: str*, *message_format: feast.data_format.StreamFormat*, *topic: str*, *created_timestamp_column: Optional[str] = ''*, *field_mapping: Optional[Dict[str, str]] = {}*, *date_partition_column: Optional[str] = ''*)

property kafka_options
Returns the kafka options of this data source

to_proto() → feast.core.DataSource_pb2.DataSource
Converts an DataSourceProto object to its protobuf representation.

class feast.data_source.**KinesisOptions**(*record_format: feast.data_format.StreamFormat*, *region: str*, *stream_name: str*)
DataSource Kinesis options used to source features from Kinesis records

classmethod from_proto(*kinesis_options_proto: feast.core.DataSource_pb2.KinesisOptions*)
Creates a KinesisOptions from a protobuf representation of a kinesis option

Parameters **kinesis_options_proto** – A protobuf representation of a DataSource

Returns Returns a KinesisOptions object based on the kinesis_options protobuf

property record_format
Returns the data format used to encode the feature data in the Kinesis records.

property region
Returns the AWS region of Kinesis stream

property stream_name
Returns the Kinesis stream name to obtain feature data from

to_proto() → feast.core.DataSource_pb2.KinesisOptions
Converts an KinesisOptionsProto object to its protobuf representation.

Returns KinesisOptionsProto protobuf

class feast.data_source.**KinesisSource**(*event_timestamp_column: str*, *created_timestamp_column: str*, *record_format: feast.data_format.StreamFormat*, *region: str*, *stream_name: str*, *field_mapping: Optional[Dict[str, str]] = {}*, *date_partition_column: Optional[str] = ''*)

property kinesis_options
Returns the kinesis options of this data source

to_proto() → feast.core.DataSource_pb2.DataSource
Converts an DataSourceProto object to its protobuf representation.

class feast.data_source.**SourceType**(*value*)
DataSource value type. Used to define source types in DataSource.

CHAPTER

THREE

ENTITY

class feast.entity.**Entity**(*name: str*, *description: str*, *value_type: feast.value_type.ValueType*, *labels: Optional[MutableMapping[str, str]] = None*)

Represents a collection of entities and associated metadata.

property created_timestamp

Returns the created_timestamp of this entity

property description

Returns the description of this entity

classmethod from_dict(*entity_dict*)

Creates an entity from a dict

Parameters **entity_dict** – A dict representation of an entity

Returns Returns a EntityV2 object based on the entity dict

classmethod from_proto(*entity_proto: feast.core.Entity_pb2.Entity*)

Creates an entity from a protobuf representation of an entity

Parameters **entity_proto** – A protobuf representation of an entity

Returns Returns a EntityV2 object based on the entity protobuf

classmethod from_yaml(*yml: str*)

Creates an entity from a YAML string body or a file path

Parameters **yml** – Either a file path containing a yaml file or a YAML string

Returns Returns a EntityV2 object based on the YAML file

is_valid()

Validates the state of a entity locally. Raises an exception if entity is invalid.

property labels

Returns the labels of this entity. This is the user defined metadata defined as a dictionary.

property last_updated_timestamp

Returns the last_updated_timestamp of this entity

property name

Returns the name of this entity

to_dict() → Dict

Converts entity to dict

Returns Dictionary object representation of entity

to_proto() → feast.core.Entity_pb2.Entity

Converts an entity object to its protobuf representation

Returns EntityV2Proto protobuf

`to_spec_proto()` → feast.core.Entity_pb2.EntitySpecV2
Converts an EntityV2 object to its protobuf representation. Used when passing EntitySpecV2 object to Feast request.

Returns EntitySpecV2 protobuf

`to_yaml()`
Converts a entity to a YAML string.

Returns Entity string returned in YAML format

`property value_type`
Returns the type of this entity

CHAPTER

FOUR

FEATURE TABLE

class feast.feature_table.**FeatureTable**(*name: str*, *entities: List[str]*, *features: List[feast.feature.Feature]*, *batch_source: Optional[Union[feast.data_source.BigQuerySource, feast.data_source.FileSource]] = None*, *stream_source: Optional[Union[feast.data_source.KafkaSource, feast.data_source.KinesisSource]] = None*, *max_age: Optional[google.protobuf.duration_pb2.Duration] = None*, *labels: Optional[MutableMapping[str, str]] = None*)

Represents a collection of features and associated metadata.

add_feature(*feature:* feast.feature.Feature)

Adds a new feature to the feature table.

property batch_source

Returns the batch source of this feature table

property created_timestamp

Returns the created_timestamp of this feature table

property entities

Returns the entities of this feature table

property features

Returns the features of this feature table

classmethod from_dict(*ft_dict*)

Creates a feature table from a dict

Parameters **ft_dict** – A dict representation of a feature table

Returns Returns a FeatureTable object based on the feature table dict

classmethod from_proto(*feature_table_proto: feast.core.FeatureTable_pb2.FeatureTable*)

Creates a feature table from a protobuf representation of a feature table

Parameters **feature_table_proto** – A protobuf representation of a feature table

Returns Returns a FeatureTableProto object based on the feature table protobuf

classmethod from_yaml(*yml: str*)

Creates a feature table from a YAML string body or a file path

Parameters **yml** – Either a file path containing a yaml file or a YAML string

Returns Returns a FeatureTable object based on the YAML file

is_valid()

Validates the state of a feature table locally. Raises an exception if feature table is invalid.

property labels
Returns the labels of this feature table. This is the user defined metadata defined as a dictionary.

property last_updated_timestamp
Returns the last_updated_timestamp of this feature table

property max_age
Returns the maximum age of this feature table. This is the total maximum amount of staleness that will be allowed during feature retrieval for each specific feature that is looked up.

property name
Returns the name of this feature table

property stream_source
Returns the stream source of this feature table

to_dict() → Dict
Converts feature table to dict

Returns Dictionary object representation of feature table

to_proto() → feast.core.FeatureTable_pb2.FeatureTable
Converts an feature table object to its protobuf representation

Returns FeatureTableProto protobuf

to_spec_proto() → feast.core.FeatureTable_pb2.FeatureTableSpec
Converts an FeatureTableProto object to its protobuf representation. Used when passing FeatureTableSpecProto object to Feast request.

Returns FeatureTableSpecProto protobuf

to_yaml()
Converts a feature table to a YAML string.

Returns Feature table string returned in YAML format

CHAPTER

FIVE

FEATURE

class feast.feature.**Feature**(*name: str*, *dtype: feast.value_type.ValueType*, *labels: Optional[MutableMapping[str, str]] = None*)

Feature field type

property dtype: feast.value_type.ValueType

Getter for data type of this field

classmethod from_proto(*feature_proto: feast.core.Feature_pb2.FeatureSpecV2*)

Parameters **feature_proto** – FeatureSpecV2 protobuf object

Returns Feature object

property labels: MutableMapping[str, str]

Getter for labels of this field

property name

Getter for name of this field

to_proto() → feast.core.Feature_pb2.FeatureSpecV2

Converts Feature object to its Protocol Buffer representation

class feast.feature.**FeatureRef**(*name: str*, *feature_table: Optional[str] = None*)

Feature Reference represents a reference to a specific feature.

classmethod from_proto(*proto: feast.serving.ServingService_pb2.FeatureReferenceV2*)

Construct a feature reference from the given FeatureReference proto Arg:

proto: Protobuf FeatureReference to construct from

Returns FeatureRef that refers to the given feature

classmethod from_str(*feature_ref_str: str*)

Parse the given string feature reference into FeatureRef model String feature reference should be in the format feature_table:feature. Where "feature_table" and "name" are the feature_table name and feature name respectively. :param feature_ref_str: String representation of the feature reference

Returns FeatureRef that refers to the given feature

to_proto() → feast.serving.ServingService_pb2.FeatureReferenceV2

Convert and return this feature table reference to protobuf. :returns: Protobuf respresentation of this feature table reference.

CHAPTER

SIX

CONSTANTS

feast.constants.**CONFIG_FEAST_ENV_VAR_PREFIX: str = 'FEAST_'**
Default prefix to Feast environmental variables

feast.constants.**CONFIG_FILE_DEFAULT_DIRECTORY: str = '.feast'**
Default directory to Feast configuration file

feast.constants.**CONFIG_FILE_NAME: str = 'config'**
Default Feast configuration file name

feast.constants.**CONFIG_FILE_SECTION: str = 'general'**
Default section in Feast configuration file to specify options

class feast.constants.**ConfigOptions**
Feast Configuration Options

AUTH_PROVIDER: str = 'auth_provider'
Authentication Provider - Google OpenID/OAuth

Options: "google" / "oauth"

AUTH_TOKEN: Optional[str] = 'auth_token'
JWT Auth token for user authentication to Feast

AZURE_BLOB_ACCOUNT_ACCESS_KEY: Optional[str] = 'azure_blob_account_access_key'
Account access key for Azure blob storage_client

AZURE_BLOB_ACCOUNT_NAME: Optional[str] = 'azure_blob_account_name'
Account name for Azure blob storage_client

BATCH_FEATURE_REQUEST_WAIT_TIME_SECONDS: str = 'batch_feature_request_wait_time_seconds'
Time to wait for historical feature requests before timing out.

BATCH_INGESTION_PRODUCTION_TIMEOUT: str = 'batch_ingestion_production_timeout'
Default timeout when running batch ingestion

CORE_ENABLE_SSL: str = 'core_enable_ssl'
Enable or disable TLS/SSL to Feast Core

CORE_SERVER_SSL_CERT: str = 'core_server_ssl_cert'
Path to certificate(s) to secure connection to Feast Core

CORE_URL: str = 'core_url'
Default Feast Core URL

DATAPROC_CLUSTER_NAME: Optional[str] = 'dataproc_cluster_name'
Dataproc cluster to run Feast Spark Jobs in

DATAPROC_EXECUTOR_CORES = 'dataproc_executor_cores'
No. of executor cores for Dataproc cluster

DATAPROC_EXECUTOR_INSTANCES = 'dataproc_executor_instances'
No. of executor instances for Dataproc cluster

DATAPROC_EXECUTOR_MEMORY = 'dataproc_executor_memory'
No. of executor memory for Dataproc cluster

DATAPROC_PROJECT: Optional[str] = 'dataproc_project'
Project of Dataproc cluster

DATAPROC_REGION: Optional[str] = 'dataproc_region'
Region of Dataproc cluster

DEADLETTER_PATH: str = 'deadletter_path'
Ingestion Job DeadLetter Destination. The choice of storage is connected to the choice of SPARK_LAUNCHER.

Eg. gs://some-bucket/output/, s3://some-bucket/output/, file:///data/subfolder/

EMR_CLUSTER_ID: Optional[str] = 'emr_cluster_id'
EMR cluster to run Feast Spark Jobs in

EMR_CLUSTER_TEMPLATE_PATH: Optional[str] = 'emr_cluster_template_path'
Template path of EMR cluster

EMR_LOG_LOCATION: Optional[str] = 'emr_log_location'
Log path of EMR cluster

EMR_REGION: Optional[str] = 'emr_region'
Region of EMR cluster

ENABLE_AUTH: str = 'enable_auth'
Enable user authentication to Feast Core

GRPC_CONNECTION_TIMEOUT: str = 'grpc_connection_timeout'
Default connection timeout to Feast Serving, Feast Core, and Feast Job Service (in seconds)

GRPC_CONNECTION_TIMEOUT_APPLY: str = 'grpc_connection_timeout_apply'
Default gRPC connection timeout when sending an ApplyFeatureTable command to Feast Core (in seconds)

HISTORICAL_FEATURE_OUTPUT_FORMAT: str = 'historical_feature_output_format'
File format of historical retrieval features

HISTORICAL_FEATURE_OUTPUT_LOCATION: Optional[str] = 'historical_feature_output_location'
File location of historical retrieval features

INGESTION_DROP_INVALID_ROWS = 'ingestion_drop_invalid_rows'
If set to true rows that do not pass custom validation (see feast.contrib.validation) won't be saved to Online Storage

JOB_SERVICE_ENABLE_CONTROL_LOOP: str = 'job_service_enable_control_loop'
Enable or disable control loop for Feast Job Service

JOB_SERVICE_ENABLE_SSL: str = 'job_service_enable_ssl'
Enable or disable TLS/SSL to Feast Job Service

JOB_SERVICE_SERVER_SSL_CERT: str = 'job_service_server_ssl_cert'
Path to certificate(s) to secure connection to Feast Job Service

JOB_SERVICE_URL: Optional[str] = 'job_service_url'
Default Feast Job Service URL

OAUTH_AUDIENCE: Optional[str] = 'oauth_audience'
Oauth intended recipients

OAUTH_CLIENT_ID: Optional[str] = 'oauth_client_id'
Oauth client ID

OAUTH_CLIENT_SECRET: Optional[str] = 'oauth_client_secret'
Oauth client secret

OAUTH_GRANT_TYPE: Optional[str] = 'oauth_grant_type'
Oauth grant type

OAUTH_TOKEN_REQUEST_URL: Optional[str] = 'oauth_token_request_url'
Oauth token request url

PROJECT: str = 'project'
Feast project namespace to use

REDIS_HOST: str = 'redis_host'
Default Redis host

REDIS_PORT: str = 'redis_port'
Default Redis port

REDIS_SSL: str = 'redis_ssl'
Enable or disable TLS/SSL to Redis

S3_ENDPOINT_URL: Optional[str] = 's3_endpoint_url'
Endpoint URL for S3 storage_client

SERVING_ENABLE_SSL: str = 'serving_enable_ssl'
Enable or disable TLS/SSL to Feast Serving

SERVING_SERVER_SSL_CERT: str = 'serving_server_ssl_cert'
Path to certificate(s) to secure connection to Feast Serving

SERVING_URL: str = 'serving_url'
Default Feast Serving URL

SPARK_BQ_MATERIALIZATION_DATASET: Optional[str] = 'spark_bq_materialization_dataset'
The dataset id where the materialized view of BigQuerySource is going to be created by default, use the same dataset where view is located

SPARK_BQ_MATERIALIZATION_PROJECT: Optional[str] = 'spark_bq_materialization_project'
The project id where the materialized view of BigQuerySource is going to be created by default, use the same project where view is located

SPARK_HOME: Optional[str] = 'spark_home'
Directory where Spark is installed

SPARK_INGESTION_JAR: str = 'spark_ingestion_jar'
Feast Spark Job ingestion jar file. The choice of storage is connected to the choice of SPARK_LAUNCHER.

Eg. “dataproc” (http and gs), “emr” (http and s3), “standalone” (http and file)

SPARK_LAUNCHER: Optional[str] = 'spark_launcher'
Spark Job launcher. The choice of storage is connected to the choice of SPARK_LAUNCHER.

Options: “standalone”, “dataproc”, “emr”

SPARK_STAGING_LOCATION: Optional[str] = 'spark_staging_location'
Feast Spark Job ingestion jobs staging location. The choice of storage is connected to the choice of SPARK_LAUNCHER.

Eg. gs://some-bucket/output/, s3://some-bucket/output/, file:///data/subfolder/

SPARK_STANDALONE_MASTER: str = 'spark_standalone_master'
Spark resource manager master url

STATSD_ENABLED: str = 'statsd_enabled'
Enable or disable StatsD

STATSD_HOST: Optional[str] = 'statsd_host'
Default StatsD port

STATSD_PORT: Optional[str] = 'statsd_port'
Default StatsD port

STENCIL_URL: str = 'stencil_url'
ProtoRegistry Address (currently only Stencil Server is supported as registry) https://github.com/gojekfarm/stencil

TELEMETRY = 'telemetry'
Telemetry enabled

feast.constants.**DATETIME_COLUMN: str = 'datetime'**
Default datetime column name for point-in-time join

feast.constants.**FEAST_CONFIG_FILE_ENV: str = 'FEAST_CONFIG'**
Environmental variable to specify Feast configuration file location

PYTHON MODULE INDEX

f

INDEX

K

L

M

N

O

P

R

S

T

V