Documentation > FeatureStore 0.6.0 documentation

Databricks Feature Store

Python API

Databricks FeatureStoreClient

class databricks.feature_store.client.FeatureStoreClient(feature_store_uri: Optional[str] = None,
model_registry_uri: Optional[str] = None)

Bases: object

Client for interacting with the Databricks Feature Store.

 $create_table(name: str, primary_keys: Union[str, List[str]], df: Optional[pyspark.sql.dataframe.DataFrame] = None, *, timestamp_keys: Union[str, List[str], None] = None, partition_columns: Union[str, List[str], None] = None, schema: Optional[pyspark.sql.types.StructType] = None, description: Optional[str] = None, tags: Optional[Dict[str, str]] = None, **kwargs) <math>\rightarrow$ databricks.feature_store.entities.feature_table.FeatureTable

Create and return a feature table with the given name and primary keys.

The returned feature table has the given name and primary keys. Uses the provided **schema** or the inferred schema of the provided **af**. If **af** is provided, this data will be saved in a Delta table. Supported data types for features are:

IntegerType, LongType, FloatType, DoubleType, StringType, BooleanType, DateType, TimestampType, ShortType, ArrayType, MapType, and BinaryType, and DecimalType.

- Parameters: name A feature table name of the form <database name>., for example dev.user_features.
 - primary_keys The feature table's primary keys. If multiple columns are required, specify a list of column names, for example ['customer id', 'region'].
 - df Data to insert into this feature table. The schema of df will be used as the feature table schema.
 - timestamp_keys -

Columns containing the event time associated with feature value. Timestamp keys and primary keys of the feature table uniquely identify the feature value for an entity at a point in time.

Note

Experimental: This argument may change or be removed in a future release without warning.

partition_columns -

Columns used to partition the feature table. If a list is provided, column ordering in the list will be used for partitioning.

Note

When choosing partition columns for your feature table, use columns that do not have a high cardinality. An ideal strategy would be such that you expect data in each partition to be at least 1 GB. The most commonly used partition column is a date.

Additional info: Choosing the right partition columns for Delta tables

- schema Feature table schema. Either schema or df must be provided.
- **description** Description of the feature table.
- tags –

Tags to associate with the feature table.

Note

Available in version >= 0.4.1.

Other Parameters:

• path (optional[str]) - Path in a supported filesystem. Defaults to the database location.

 $\textbf{register_table}(\ ^*, \ delta_table: str, \ primary_keys: \ Union[str, \ List[str]], \ timestamp_keys: \ Union[str, \ List[str], \ None] = None, \ timestamp_keys: \ Union[str, \ List[str]], \ timestamp_keys: \ Union[str]], \ timestamp_keys: \ Union[str, \ List[str]], \ timestamp_keys: \ Union[str]], \ timestamp_keys: \ Union$ description: Optional[str] = None, tags: Optional[Dict[str, str]] = None) \rightarrow databricks.feature_store.entities.feature_table.FeatureTable

Register an existing Delta table as a feature table with the given primary keys.

The returned feature table has the same name as the Delta table.

Note

Available in version >= 0.3.8.

- Parameters: name A Delta table name of the form <database_name>.<table_name>, for example dev.user_features. The table must exist in the metastore.
 - primary_keys The Delta table's primary keys. If multiple columns are required, specify a list of column names, for example ['customer_id', 'region'].
 - timestamp_keys Columns containing the event time associated with feature value. Together, the timestamp keys and primary keys uniquely identify the feature value at a point in time.
 - **description** Description of the feature table.

Tags to associate with the feature table.

Note

Available in version >= 0.4.1.

A FeatureTable object. Returns:

 $\texttt{get_table}(\textit{name: str}) \rightarrow \texttt{databricks.feature_store.entities.feature_table.FeatureTable}$

Get a feature table's metadata.

Parameters: name - A feature table name of the form <database name>., for example dev.user features.

drop table(name: str) → None

Note

Experimental: This method may change or be removed in a future release without warning.

Delete the specified feature table. This API also drops the underlying Delta table.

Note

Available in version >= 0.4.1.

Parameters: name - The feature table name of the form <database_name>.<table_name>, for example dev.user_features.

Note

Deleting a feature table can lead to unexpected failures in upstream producers and downstream consumers (models, endpoints, and scheduled jobs). You must delete any existing published online stores separately.

read table(name: str, **kwargs) → pyspark.sql.dataframe.DataFrame

Read the contents of a feature table.

Parameters: name - A feature table name of the form <database_name>.<table_name>, for example dev.user_features.

The feature table contents, or an exception will be raised if this feature table does not exist. Returns:

write table(name: str, df: pyspark.sql.dataframe.DataFrame, mode: str = 'merge', checkpoint location: Optional[str] = None, trigger: Dict[str, Any] = {'processingTime': '5 seconds'}} → Optional[pyspark.sql.streaming.StreamingQuery]

Writes to a feature table.

If the input **DataFrame** is streaming, will create a write stream.

- Parameters: name A feature table name of the form <database_name>.<table_name>, for example dev.user_features. Raises an exception if this feature table does not exist.
 - df Spark DataFrame with feature data. Raises an exception if the schema does not match that of the feature table.
 - mode –

Two supported write modes:

- "overwrite" updates the whole table.
- o "merge" will upsert the rows in df into the feature table. If df contains columns not present in the feature table, these columns will be added as new features.
- checkpoint_location Sets the Structured Streaming checkpointLocation option. By setting a checkpoint_location, Spark Structured Streaming will store progress information and intermediate state, enabling recovery after failures. This parameter is only supported when the argument df is a streaming DataFrame.
- trigger If df.isStreaming, trigger defines the timing of stream data processing, the dictionary will be unpacked and passed to DataStreamWriter.trigger as arguments. For example,

trigger={'once': True} will result in a call to DataStreamWriter.trigger(once=True).

If df.isStreaming, returns a PySpark StreamingQuery. None otherwise. Returns:

add data sources(*, feature_table_name: str, source_names: Union[str, List[str]], source_type: str = 'custom') → None

Note

Experimental: This method may change or be removed in a future release without warning.

Add data sources to the feature table.

- **Parameters:** feature_table_name The feature table name.
 - source_names Data source names. For multiple sources, specify a list. If a data source name already exists, it is ignored.
 - source_type -One of the following:
 - "table": Table in format <database_name>.<table_name> and is stored in the metastore (eg Hive).
 - o "path": Path, eg in the Databricks File System (DBFS).
 - o "custom": Manually added data source, neither a table nor a path.

 $\mathtt{delete_data_sources}(\texttt{*}, feature_table_name: str, source_names: Union[str, List[str]]) \rightarrow \mathsf{None}(\texttt{str}, \texttt{source})$

Note

Experimental: This method may change or be removed in a future release without warning.

Delete data sources from the feature table.



- **Parameters:** feature_table_name The feature table name.
 - source_names Data source names. For multiple sources, specify a list. If a data source name does not exist, it is ignored.

publish table(name: str, online_store: databricks.feature_store.online_store_spec.online_store_spec.OnlineStoreSpec, *, filter_condition: Optional[str] = None, mode: str = 'merge', streaming: bool = False, checkpoint_location: Optional[str] = None, $trigger: Dict[str, Any] = \{'processingTime': '5 minutes'\}, features: Union[str, List[str], None] = None\} \rightarrow$ Optional[pyspark.sql.streaming.StreamingQuery]

Publish a feature table to an online store.

- Parameters: name Name of the feature table.
 - online_store Specification of the online store.
 - filter_condition A SQL expression using feature table columns that filters feature rows prior to publishing to the online store. For example, "dt > '2020-09-10'". This is analogous to running df.filter or a WHERE condition in SQL on a feature table prior to publishing.

Specifies the behavior when data already exists in this feature table in the online store. If "overwrite" mode is used, existing data is replaced by the new data. If "merge" mode is used, the new data will be merged in, under these conditions:

- If a key exists in the online table but not the offline table, the row in the online table is unmodified.
- If a key exists in the offline table but not the online table, the offline table row is inserted into the online
- If a key exists in both the offline and the online tables, the online table row will be updated.
- **streaming** If **True**, streams data to the online store.
- checkpoint_location Sets the Structured Streaming checkpointLocation option. By setting a **checkpoint** location, Spark Structured Streaming will store progress information and intermediate state, enabling recovery after failures. This parameter is only supported when **streaming=True**.
- trigger If streaming=True, trigger defines the timing of stream data processing. The dictionary will be unpacked and passed to DataStreamWriter.trigger as arguments. For example,

trigger={'once': True} will result in a call to DataStreamWriter.trigger(once=True).

features –

Specifies the feature column(s) to be published to the online store. The selected features must be a superset of existing online store features. Primary key columns and timestamp key columns will always be published.

Note

This parameter is only supported when mode="merge". When features is not set, the whole feature table will be published.

If **streaming=True**, returns a PySpark **StreamingQuery**, **None** otherwise. Returns:

create training set(df: pyspark.sql.dataframe.DataFrame, feature lookups: List[databricks.feature_store.entities.feature_lookup.FeatureLookup], label: Union[str, List[str], None], exclude_columns: *List[str] = []*) → databricks.feature_store.training_set.TrainingSet

Parameters: • **df** – The **DataFrame** used to join features into.

- feature_lookups List of features to join into the DataFrame.
- label Names of column(s) in DataFrame that contain training set labels. To create a training set without a label field, i.e. for unsupervised training set, specify label = None.
- exclude_columns Names of the columns to drop from the TrainingSet DataFrame.

A **TrainingSet** object. Returns:

log_model(model: Any, artifact_path: str, *, flavor: module, training_set: databricks.feature_store.training_set.TrainingSet, registered_model_name: Optional[str] = None, await_registration_for: int = 300, **kwargs)

Log an MLflow model packaged with feature lookup information.

Note

The DataFrame returned by TrainingSet.load df() must be used to train the model. If it has been modified (for example data normalization, add a column, and similar), these modifications will not be applied at inference time, leading to training-serving skew.

- Parameters: model Model to be saved. This model must be capable of being saved by flavor.save_model. See the MLflow Model API.
 - artifact_path Run-relative artifact path.
 - flavor MLflow module to use to log the model. flavor should have type ModuleType. The module must have a method **save_model**, and must support the **python_function** flavor. For example, mlflow.sklearn, mlflow.xgboost, and similar.
 - training_set The TrainingSet used to train this model.
 - registered_model_name -

Note

Experimental: This argument may change or be removed in a future release without warning.

If given, create a model version under registered model name, also creating a registered model if one with the given name does not exist.

 await_registration_for – Number of seconds to wait for the model version to finish being created and is in **READY** status. By default, the function waits for five minutes. Specify **0** or **None** to skip waiting.

None Returns:

score batch(model_uri: str, df: pyspark.sql.dataframe.DataFrame, result_type: str = 'double') → pyspark.sql.dataframe.DataFrame

Evaluate the model on the provided **DataFrame**.

Additional features required for model evaluation will be automatically retrieved from Feature Store.

The model must have been logged with FeatureStoreClient.log model(), which packages the model with feature metadata. Unless present in df, these features will be looked up from Feature Store and joined with df prior to scoring the model.

If a feature is included in **df**, the provided feature values will be used rather than those stored in **Feature Store**.

For example, if a model is trained on two features account_creation_date and num_lifetime_purchases, as in:

```
feature_lookups = [
    FeatureLookup(
        table_name = 'trust_and_safety.customer_features',
        feature_name = 'account_creation_date',
        lookup_key = 'customer_id',
    ),
    FeatureLookup(
        table_name = 'trust_and_safety.customer_features',
        feature_name = 'num_lifetime_purchases',
       lookup_key = 'customer_id'
   ),
]
with mlflow.start run():
    training_set = fs.create_training_set(
        feature lookups = feature lookups,
       label = 'is_banned',
        exclude_columns = ['customer_id']
    )
     fs.log model(
       model.
        "model",
       flavor=mlflow.sklearn,
       training_set=training_set,
       registered model name="example model"
      )
```

Then at inference time, the caller of FeatureStoreClient.score_batch() must pass a DataFrame that includes customer_id, the lookup_key specified in the FeatureLookups of the training_set. If the DataFrame contains a column account_creation_date, the values of this column will be used in lieu of those in Feature Store. As in:

```
# batch_df has columns ['customer_id', 'account_creation_date']
predictions = fs.score_batch(
    'models:/example_model/1',
    batch_df
)
```

Parameters: • model_uri -

The location, in URI format, of the MLflow model logged using **FeatureStoreClient.log_model()**. One of:

- o runs:/<mlflow run id>/run-relative/path/to/model
- o models:/<model_name>/<model_version>
- o models:/<model_name>/<stage>

For more information about URI schemes, see Referencing Artifacts.

df -

The **DataFrame** to score the model on. **Feature Store** features will be joined with **df** prior to scoring the model. **df** must:

- 1. Contain columns for lookup keys required to join feature data from Feature Store, as specified in the **feature spec.yaml** artifact.
- 2. Contain columns for all source keys required to score the model, as specified in the **feature_spec.yam1** artifact.
- 3. Not contain a column **prediction**, which is reserved for the model's predictions. **df** may contain additional columns.
- result_type The return type of the model. See mlflow.pyfunc.spark_udf() result_type.

Returns:

A **DataFrame** containing:

- 1. All columns of df.
- 2. All feature values retrieved from Feature Store.
- 3. A column **prediction** containing the output of the model.

set feature table tag(*, table_name: str, key: str, value: str) → None

Create or update a tag associated with the feature table. If the tag with the corresponding key already exists, its value will be overwritten with the new value.

Note

Available in version >= 0.4.1.

Parameters:

- table_name the feature table name
- key tag key
- value tag value

delete_feature_table_tag(*, table_name: str, key: str) → None

Delete the tag associated with the feature table. Deleting a non-existent tag will emit a warning.

Note

Available in version >= 0.4.1.

Parameters:

- table_name the feature table name.
- **key** the tag key to delete.

Feature Lookup

class databricks.feature store.entities.feature lookup.FeatureLookup(table_name: str, lookup_key: Union[str, List[str]], *, feature_names: Union[str, List[str], None] = None, rename_outputs: Optional[Dict[str, str]] = None, timestamp_lookup_key: Union[str, List[str], None] = None, **kwargs)

Bases: databricks.feature_store.entities._feature_store_object._FeatureStoreObject

Value class used to specify a feature to use in a **TrainingSet**.

- **Parameters:** table_name Feature table name.
 - lookup_key Key to use when joining this feature table with the DataFrame passed to FeatureStoreClient.create_training_set(). The lookup_key must be the columns in the DataFrame passed to FeatureStoreClient.create_training_set(). The type of lookup_key columns in that DataFrame must match the type of the primary key of the feature table referenced in this FeatureLookup.
 - feature_names A single feature name, a list of feature names, or None to lookup all features (excluding primary keys) in the feature table at the time that the training set is created. If your model requires primary keys as features, you can declare them as independent FeatureLookups.
 - rename_outputs If provided, renames features in the TrainingSet returned by of FeatureStoreClient.create_training_set.
 - timestamp_lookup_key -Key to use when performing point-in-time lookup on this feature table with the DataFrame passed to FeatureStoreClient.create_training_set(). The timestamp_lookup_key must be the columns in the DataFrame passed to FeatureStoreClient.create training set(). The type of timestamp_lookup_key columns in that DataFrame must match the type of the timestamp key of the feature table referenced in this **FeatureLookup**.

Note

Experimental: This argument may change or be removed in a future release without warning.

- feature_name Feature name. Deprecated as of version 0.3.4. Use feature names.
- output_name If provided, rename this feature in the output of FeatureStoreClient.create_training_set. Deprecated as of version 0.3.4. Use rename_outputs.

init (table_name: str, lookup_key: Union[str, List[str]], *, feature_names: Union[str, List[str], None] = None, rename_outputs: Optional[Dict[str, str]] = None, timestamp_lookup_key: Union[str, List[str], None] = None, **kwargs)

Initialize a FeatureLookup object.

table name

The table name to use in this FeatureLookup.

lookup key

The lookup key(s) to use in this FeatureLookup.

feature name

The feature name to use in this FeatureLookup. **Deprecated** as of version 0.3.4. Use **feature_names**.

output name

The output name to use in this FeatureLookup. **Deprecated** as of version 0.3.4. Use **feature names**.

Training Set

class databricks.feature store.training set.TrainingSet(feature_spec:

databricks.feature_store.entities.feature_spec.FeatureSpec, df: pyspark.sql.dataframe.DataFrame, labels: List[str], feature_table_metadata_map: Dict[str, databricks.feature_store.entities.feature_table.FeatureTable], feature_table_data_map: Dict[str, pyspark.sql.dataframe.DataFrame])

Bases: object

Class that defines **TrainingSet** objects.

Note

The **TrainingSet** constructor should not be called directly. Instead, call **FeatureStoreClient.create_training_set**.

load_df() → pyspark.sql.dataframe.DataFrame

Load a DataFrame.

Return a DataFrame for training.

The returned **DataFrame** has columns specified in the **feature_spec** and **labels** parameters provided in **FeatureStoreClient.create_training_set**.

Returns:

A DataFrame for training

Feature Table

Classes

class databricks.feature_store.entities.feature_table.FeatureTable(name, table_id, description, primary_keys, partition_columns, features, creation_timestamp=None, online_stores=[], notebook_producers=[], table_data_sources=[], path_data_sources=[], custom_data_sources=[], timestamp_keys=[], tags=None)

Value class describing one feature table.

This will typically not be instantiated directly, instead the **FeatureStoreClient.create_table** will create **FeatureTable** objects.

Online Store Spec

class databricks.feature_store.online_store_spec.AmazonRdsMySqlSpec(hostname: str, port: int, user:

Optional[str] = None, password: Optional[str] = None, database_name: Optional[str] = None, table_name: Optional[str] = None, driver_name: Optional[str] = None, read_secret_prefix: Optional[str] = None, write_secret_prefix: Optional[str] = None)

Bases: databricks.feature_store.online_store_spec.online_store_spec.OnlineStoreSpec

Class that defines and creates AmazonRdsMySqlSpec objects.

This **OnlineStoreSpec** implementation is intended for publishing features to Amazon RDS MySQL and Aurora (MySQL-compatible edition).

See OnlineStoreSpec documentation for more usage information, including parameter descriptions.

Parameters:

- hostname Hostname to access online store.
- port Port number to access online store.
- user Username that has access to the online store. Deprecated as of version 0.6.0. Use
 write_secret_prefix instead.
- password Password to access the online store. Deprecated as of version 0.6.0. Use write_secret_prefix instead.
- database_name Database name.
- table_name Table name.
- driver_name Name of custom JDBC driver to access the online store.
- read_secret_prefix Prefix for read secret.
- write_secret_prefix Prefix for write secret.

hostname

Hostname to access the online store.

port

Port number to access the online store.

database_name

Database name.

cloud

Define the cloud propert for the data store.

store type

Define the data store type property.

auth_type()

Publish Auth type.

class databricks.feature_store.online_store_spec.AzureMySqlSpec(hostname: str, port: int, user: Optional[str] = None, password: Optional[str] = None, database_name: Optional[str] = None, table_name: Optional[str] = None, driver_name: Optional[str] = None, read_secret_prefix: Optional[str] = None, write_secret_prefix: Optional[str] = None)

Bases: databricks.feature_store.online_store_spec.online_store_spec.OnlineStoreSpec

Define the AzureMySqlSpec class.

This **OnlineStoreSpec** implementation is intended for publishing features to Azure Database for MySQL.

See OnlineStoreSpec documentation for more usage information, including parameter descriptions.

Parameters:

- hostname Hostname to access online store.
- port Port number to access online store.
- user Username that has access to the online store. Deprecated as of version 0.6.0. Use
 write_secret_prefix instead.
- password Password to access the online store. Deprecated as of version 0.6.0. Use write_secret_prefix instead.
- database_name Database name.
- table_name Table name.
- driver_name Name of custom JDBC driver to access the online store.
- read_secret_prefix Prefix for read secret.
- write_secret_prefix Prefix for write secret.

hostname

Hostname to access the online store.

port

Port number to access the online store.

database_name

Database name.

cloud

Define the cloud the fature store runs.

store_type

Define the data store type.

auth_type()

Publish Auth type.

class $databricks.feature_store.online_store_spec.AzureSqlServerSpec(hostname: str, port: int, user: Optional[str] = None, password: Optional[str] = None, database_name: Optional[str] = None, table_name: Optional[str] = None, driver_name: Optional[str] = None, read_secret_prefix: Optional[str] = None, write_secret_prefix: Optional[str] = None)$

Bases: databricks.feature_store.online_store_spec.online_store_spec.OnlineStoreSpec

This **OnlineStoreSpec** implementation is intended for publishing features to Azure SQL Database (SQL Server).

The spec supports SQL Server 2019 and newer.

See **OnlineStoreSpec** documentation for more usage information, including parameter descriptions.

Parameters:

- hostname Hostname to access online store.
- port Port number to access online store.
- user Username that has access to the online store. **Deprecated** as of version 0.6.0. Use write_secret_prefix instead.
- password Password to access the online store. Deprecated as of version 0.6.0. Use write_secret_prefix instead.
- database_name Database name.
- table_name Table name.
- driver_name Name of custom JDBC driver to access the online store.
- read_secret_prefix Prefix for read secret.
- write_secret_prefix Prefix for write secret.

hostname

Hostname to access the online store.

port

Port number to access the online store.

database name

Database name.

cloud

Define the cloud the fature store runs.

store type

Define the data store type.

auth_type()

Publish Auth type.

class databricks.feature_store.online_store_spec.AmazonDynamoDBSpec(*, region: Optional[str], access_key_id: Optional[str] = None, secret_access_key: Optional[str] = None, session_token: Optional[str] = None, table_name: Optional[str] = None, read_secret_prefix: Optional[str] = None, write_secret_prefix: Optional[str] = None, ttl: Optional[datetime.timedelta] = None)

Bases: databricks.feature_store.online_store_spec.online_store_spec.OnlineStoreSpec

This **OnlineStoreSpec** implementation is intended for publishing features to Amazon DynamoDB.

If **table_name** is not provided, **FeatureStoreClient.publish_table** will use the offline store's database and table name combined as the online table name.

To use a different table name in the online store, provide a value for the table name argument.

The expected read or write secrets for DynamoDB for a given {prefix} string are \${prefix}-access-key-id, \${prefix}-secret-access-key, and \${prefix}-session-token.

If none of the access_key_id, secret_access_key, and write_secret_prefix are passed, the instance profile attached to the cluster will be used to write to DynamoDB.

Note

AmazonDynamoDBSpec is available in version >= 0.3.8.

- **Parameters:** region Region to access online store.
 - access_key_id Access key ID that has access to the online store. Deprecated as of version 0.6.0. Use write secret prefix instead.
 - secret_access_key Secret access key to access the online store. Deprecated as of version 0.6.0. Use write_secret_prefix instead.
 - session_token Session token to access the online store. Deprecated as of version 0.6.0. Use write_secret_prefix instead.
 - table_name Table name.
 - read_secret_prefix Prefix for read secret.
 - write_secret_prefix Prefix for write secret.
 - ttl The time to live for data published to the online store. This attribute is only applicable when publishing time series feature tables. If the time to live is specified for a time series table,

FeatureStoreClient.publish_table() will publish a window of data instead of the latest snapshot.

access key id

Warning

databricks.feature_store.online_store_spec.amazon_dynamodb_online_store_spec.access_key_id is deprecated since v0.6.0. This method will be removed in a near future release. Use write secret prefix instead.

Access key ID that has access to the online store. Property will be empty if write_secret_prefix or the instance profile attached to the cluster are intended to be used.

secret_access_key

A Warning

databricks.feature_store.online_store_spec.amazon_dynamodb_online_store_spec.secret_access_key is deprecated since v0.6.0. This method will be removed in a near future release. Use write secret prefix instead.

Secret access key to access the online store. Property will be empty if write secret prefix or the instance profile attached to the cluster are intended to be used.

session_token

A Warning

databricks.feature_store.online_store_spec.amazon_dynamodb_online_store_spec.session_token is deprecated since v0.6.0. This method will be removed in a near future release. Use write_secret_prefix instead.

Session token to access the online store. Property will be empty if write secret prefix or the instance profile attached to the cluster are intended to be used.

cloud

Define the cloud property for the data store.

store_type

Define the data store type.

region

Region to access the online store.

ttl

Time to live attribute for the online store.

auth_type()

Publish Auth type.

class databricks.feature_store.online_store_spec.AzureCosmosDBSpec(*, account_uri: str, database_name: Optional[str] = None, container_name: Optional[str] = None, read_secret_prefix: Optional[str] = None, write_secret_prefix: str)

Bases: databricks.feature_store.online_store_spec.online_store_spec.OnlineStoreSpec

This **OnlineStoreSpec** implementation is intended for publishing features to Azure Cosmos DB.

If database_name and container_name are not provided, FeatureStoreClient.publish_table will use the offline store's database and table name as the Cosmos DB database and container name.

The expected read or write secret for Cosmos DB for a given {prefix} string is \${prefix}-authorization-key.

The authorization key can be either the Cosmos DB account primary or secondary key.

Note

Available in version >= 0.5.0.

Parameters:

- account_uri URI of the Cosmos DB account.
- database_name Database name.
- container_name Container name.
- read_secret_prefix Prefix for read secret.
- write_secret_prefix Prefix for write secret.

account_uri

Account URI of the online store.

database_name

Database name.

container_name

Container name.

cloud

Define the cloud property for the data store.

store_type

Define the data store type.

auth_type()

Publish Auth type.

class databricks.feature_store.online_store_spec.OnlineStoreSpec(_type, hostname: [<class 'str'>, None] =
None, port: [<class 'int'>, None] = None, user: Optional[str] = None, password: Optional[str] = None, database_name: Optional[str] =
None, table_name: Optional[str] = None, driver_name: Optional[str] = None, read_secret_prefix: Optional[str] = None,
write_secret_prefix: Optional[str] = None, _internal_properties: Optional[Dict[str, str]] = None)

Bases: abc.ABC

Parent class for all types of **OnlineStoreSpec** objects.

Abstract base class for classes that specify the online store to publish to.

If database_name and table_name are not provided, FeatureStoreClient.publish_table will use the offline store's database and table names.

To use a different database and table name in the online store, provide values for both **database_name** and **table name** arguments.

The JDBC driver can be customized with the optional driver_name argument. Otherwise, a default is used.

Strings in the primary key should not exceed 100 characters.

The online database should already exist.

Note

It is strongly suggested (but not required), to provide read-only database credentials via the read_secret_prefix
in order to grant the least amount of database access privileges to the served model. When providing a read_secret_prefix, the secrets must exist in the scope name using the expected format, otherwise publish_table will return an error.

- Parameters: hostname Hostname to access online store. The database hostname cannot be changed. Subsequent publish calls to the same online store must provide the same hostname.
 - port Port number to access online store. The database port cannot be changed. Subsequent publish calls to the same online store must provide the same port.
 - user Username that has write access to the online store. Deprecated as of version 0.6.0. Use write secret prefix instead.
 - password Password to access the online store. Deprecated as of version 0.6.0. Use write secret prefix instead
 - database name Database name.
 - table_name Table name.
 - driver_name Name of custom JDBC driver to access the online store.
 - read_secret_prefix -

The secret scope name and secret key name prefix where read-only online store credentials are stored. These credentials will be used during online feature serving to connect to the online store from the served model. The format of this parameter should be **\${scope-name}/\${prefix}**, which is the name of the secret scope, followed by a /, followed by the secret key name prefix. The scope passed in must contain the following keys and corresponding values:

- \${prefix}-user where \${prefix} is the value passed into this function. For example if this function is called with datascience/staging, the datascience secret scope should contain the secret named staging-user, which points to a secret value with the database username for the online store.
- \${prefix}-password where \${prefix} is the value passed into this function. For example if this function is called with datascience/staging, the datascience secret scope should contain the secret named staging-password, which points to a secret value with the database password for the online

Once the read_secret_prefix is set for an online store, it cannot be changed.

write_secret_prefix -

The secret scope name and secret key name prefix where read-write online store credentials are stored. These credentials will be used to connect to the online store to publish features. If user and password are passed, this field must be None, or an exception will be raised. The format of this parameter should be \${scope-name}/\${prefix}, which is the name of the secret scope, followed by a /, followed by the secret key name prefix. The scope passed in must contain the following keys and corresponding values:

- \${prefix}-user where \${prefix} is the value passed into this function. For example if this function is called with datascience/staging, the datascience secret scope should contain the secret named staging-user, which points to a secret value with the database username for the online store.
- \${prefix}-password where \${prefix} is the value passed into this function. For example if this function is called with datascience/staging, the datascience secret scope should contain the secret named staging-password, which points to a secret value with the database password for the online store.

type

Type of the online store.

table name

Table name.

user

A Warning

databricks.feature store.online store spec.online store spec.user is deprecated since v0.6.0. This method will be removed in a near future release. Use write_secret_prefix instead.

Username that has access to the online store.

Property will be empty if write_secret_prefix argument was used.

password

A Warning

databricks.feature_store.online_store_spec.online_store_spec.password is deprecated since v0.6.0. This method will be removed in a near future release. Use write_secret_prefix instead.

Password to access the online store.

Property will be empty if write secret prefix argument was used.

driver

Name of the custom JDBC driver to access the online store.

read_secret_prefix

Prefix for read access to online store.

Name of the secret scope and prefix that contains the username and password to access the online store with readonly credentials.

See the **read secret prefix** parameter description for details.

write_secret_prefix

Secret prefix that contains online store login info.

Name of the secret scope and prefix that contains the username and password to access the online store with read/write credentials. See the **write_secret_prefix** parameter description for details.

cloud

Cloud provider where this online store is located.

store_type

Store type.

auth_type()

Publish Auth type.