Apply Expectations to New Data

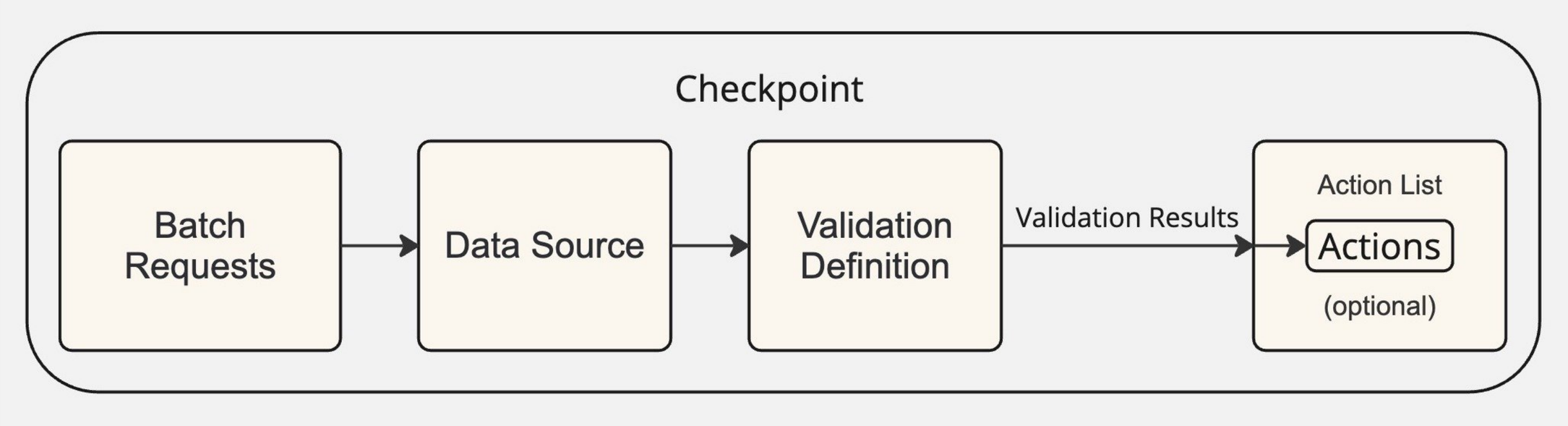
INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Checkpoints

Checkpoint - An object that groups and runs Validation Definitions with shared parameters



Actions - Components configured by Checkpoints that integrate GX with other tools based on Validation Results

1 https://docs.greatexpectations.io/docs/core/trigger_actions_based_on_results/create_a_checkpoint_with_actio

Why use Checkpoints?

Reusability

- Can run multiple Validation Definitions against a Batch

Actions

- Can trigger Actions based on Validation Results

Creating a Checkpoint

Creating a Checkpoint with Slack Notification via `gx.Checkpoint()`:

```
checkpoint = gx.Checkpoint(
    name="my_checkpoint",
    validation_definitions=[validation_definition],
    actions=[SlackNotificationAction()]  # optional
)
```

Checkpoint errors

Running a Checkpoint before adding the Validation Definition to the Data Context raises an error:

```
CheckpointRelatedResourcesFreshnessError:
ValidationDefinition 'my_validation_definition' must be added to the DataContext
before it can be updated. Please call `context.validation_definitions.add(
<VALIDATION_DEFINITION_OBJECT>)`, then try your action again.
```

Adding a Validation Definition

Add the Validation Definition to the Data Context using `.validation_definitions.add()`:

```
validation_definition = context.validation_definitions.add(
    validation_definition=validation_definition
)
```

Running a Checkpoint

```
checkpoint_results = checkpoint.run(
    batch_parameters={"dataframe": dataframe}
)
```


```
{
  "validation_result_url": "https://app.greatexpectations.io/organizations/dawatek/data-assets/12f2fd76-d127-4517-83e6-464b0567e2fb/validations/expectation-suites/0c34304b-0ce0-4b39-8f6d-9cd9b4b0e151/results/5bc5404d-872f-4804-8bfd-f43fe04333f6",
  "run_id": {
    "run_name": null,
    "run_time": "2024-06-05T14:53:12.870646-07:00"
  },
  "run_results": {
    "GXCloudIdentifier::GXCloudRESTResource.VALIDATION_RESULT::5bc5404d-872f-4804-8bfd-f43fe04333f6": {
      "validation_result": {
        "success": true,
        "results": [
          {
            "success": true,
            "expectation_config": {
              "expectation_type": "expect_column_to_exist",
              "kwargs": {
                "catch_exceptions": false,
                "column": "hero_image",
                "result_format": null,
                "batch_id": "2024-06-05 14:52:57.953264-2024-06-05 14:52:57.953264"
              }
```

Assessing Checkpoint Results

```
print(checkpoint_results.success)
```

```
False
```

```
print(checkpoint_results.describe())
```

Assessing Checkpoint Results


```
{ "success": false,
  "statistics": {
    "evaluated_expectations": 1, "successful_expectations": 0,
    "unsuccessful_expectations": 1, "success_percent": 0.0
  },
  "expectations": [{
    "expectation_type": "expect_table_row_count_to_equal",
    "success": false,
    "kwargs": {"batch_id": ""my_datasource-my_dataframe_asset", "value": 118000},
    "result": {"observed_value": 11866}}
  ],
  "result_url": "https://app.greatexpectations.io/organizations/my_org/data-assets/*
}
```

Data Docs

Data Docs - static websites generated from GX metadata

```
# Checkpoint with Action for Updating Data Docs
gx.Checkpoint(
    name,
    validation_definitions,
    actions=[
      gx.checkpoint.actions.UpdateDataDocsAction(
          name="update_my_site", site_names="my_data_docs_site"
        )
    ],
)
```

[1] https://docs.greatexpectations.io/docs/core/configure_project_settings/configure_data_docs/

Data Docs

Table level Expectations

All Expectations met

hero_image is a required field.

OBSERVED VALUE: --

Must have at least these columns (in any order): **name** **link** **price_usd** **mark_price_usd** **star_rating** **colour** **seller_name** See all

OBSERVED VALUE: ['name', 'link', 'price_usd', 'mark_price_usd', 'star_rating', 'colour', 'seller_name', 'review_count', 'sku_id', 'hero_image']

Must have greater than or equal to **1** rows.

OBSERVED VALUE: 899

Validation History

Run Time	17m ago	17m ago
Observed Value	899 ✓	899 ✓
Min Value	1	1
Max Value	None	None

Cheat sheet

Add Validation Definition to Data Context:

```
context.validation_definitions.add(
    validation_definition
)
```

Run Checkpoint:

```
checkpoint_results = checkpoint.run(
    batch_parameters={"dataframe": dataframe}
)
```

Create Checkpoint:

```
checkpoint = gx.Checkpoint(
    name: str,
    validation_definitions: list,
)
```

Check Checkpoint Results:

```
checkpoint_results.success
checkpoint_results.describe()
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Update Expectation Suites

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Adding Expectations

```
expectation = gx.expectations.ExpectTableColumnCountToEqual(
    value=10
)
```

```
suite = gx.ExpectationSuite(
    name="my_suite"
)
```

```
# Add Expectation to Suite
suite.add_expectation(
    expectation=expectation
)
```

Multiple Expectation Suites

```
# Create another Expectation Suite
another_suite = gx.ExpectationSuite(name="my_other_suite")
# Add the same Expectation to the new Suite
another_suite.add_expectation(expectation=expectation)
```

Expectations **cannot** belong to multiple Suites at once:

```
RuntimeError: Cannot add Expectation because it already belongs to an
ExpectationSuite. If you want to update an existing Expectation, please call
Expectation.save(). If you are copying this Expectation to a new ExpectationSuite,
please copy it first (the core expectations and some others support
copy(expectation)) and set `Expectation.id = None`.
```

Copying Expectations

```
If you are copying this Expectation to a new ExpectationSuite, please copy it first
(the core expectations and some others support copy(expectation)) and set
`Expectation.id = None`.
```

Copy the Expectation, set its `.id` to `None`, and add it to the new Suite without errors:

```
expectation_copy = expectation.copy()
expectation_copy.id = None
```

```
another_suite.add_expectation(
    expectation=expectation_copy
)
```

Checking Expectations

```
print(
    expectation_copy in another_suite.expectations
)
```

```
True
```

Deleting Expectations

Add

`.add_expectation()`

```
suite.add_expectation(
    expectation=expectation
)
```

Delete

`.delete_expectation()`

```
suite.delete_expectation(
    expectation=expectation
)
```

Adjusting Expectations

Update the `.value` attribute and save changes:

```
expectation = gx.expectations.ExpectTableColumnCountToEqual(
    value=10
)
expectation.value = 11
expectation.save()
```

Ensure the Expectation belongs to a Suite, otherwise:

```
RuntimeError: Expectation must be added to ExpectationSuite before it can be saved.
```

Updating an Expectation Suite

```
suite = gx.ExpectationSuite(name="my_suite")
validation_definition = gx.ValidationDefinition(
    data=batch_definition, suite=suite, name="my_validation_definition"
)
```

```
# Define Expectation
col_name_expectation = gx.expectations.ExpectColumnToExist(column="GHI")
# Add Expectation to Suite
suite.add_expectation(expectation=col_name_expectation)
# Run Validation Definition associated with the Suite
validation_results = validation_definition.run()
```

Saving an Expectation Suite

Save changes to the Suite before running the Validation Definition to avoid errors:

```
validation_results = validation_definition.run()
```

```
ResourceFreshnessAggregateError: ExpectationSuite 'my_suite' has changed since it has last been saved. Please update with `<SUITE_OBJECT>.save()`, then try your action again.
```

Saving an Expectation Suite

Use the `.save()` method to save the Suite and run the Validation Definition error-free:

```
suite.save()
validation_results = validation_definition.run()
print(validation_results.success)
```

```
False
```

Cheat sheet

Copy Expectation:

```
expectation_copy = expectation.copy()
expectation_copy.id = None
```

Check if Expectation is in Suite:

```
expectation in suite.expectations
```

Delete Expectation:

```
suite.delete_expectation(expectation)
```

Update Expectation value:

```
expectation.value = new_value
```

Save changes to Expectation:

```
expectation.save()
```

Save changes to Expectation Suite:

```
suite.save()
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Manage Components

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS

Davina Moossazadeh
Data Scientist

datacamp

Components

GX components - Python classes that represent data and data validation entities

- Data Context
- Data Sources & Data Assets
- Batch Definitions & Batches
- Expectations
- Expectation Suites
- Validation Definitions
- Checkpoints & Actions
- Data Docs

[1] https://docs.greatexpectations.io/docs/core/introduction/gx_overview

Component management in GX

Data Sources: `context.data_sources`

- connect to data and contain Data Assets

Expectation Suites: `context.suites`

- house Expectations

Validation Definitions: `context.validation_definitions`

- validate Expectations against data

Checkpoints: `context.checkpoints`

- group and automate Validations

Adding components

Expectation Suite:

```
suite = context.suites.add(suite)
```

Validation Definition:

```
validation_definition = context.validation_definitions.add(validation_definition)
```

Checkpoint:

```
checkpoint = context.checkpoints.add(
    checkpoint=checkpoint
)
```

Adding a Data Source

```
data_source = context.data_sources.add_<TYPE_NAME>()
```

	Data Source Type	Method	Parameters
0	pandas dataframe	add_pandas()	name: str
1	Spark dataframe	add_spark()	name: str
2	Local filesystem pandas	add_pandas_filesystem()	name: str, base_directory: str
3	Local filesystem Spark	add_spark_filesystem()	name: str, base_directory: str
4	Amazon S3 pandas	add_pandas_s3()	name: str, bucket: str, boto3_options: dict
5	Amazon S3 Spark	add_spark_s3()	name: str, bucket: str, boto3_options: dict
6	Azure Blob Storage pandas	add_pandas_abs()	name: str, azure_options: dict
7	Azure Blob Storage Spark	add_spark_abs()	name: str, azure_options: dict
8	Google Cloud Storage pandas	add_pandas_gcs()	name: str, bucket_or_name: str, gcs_options: dict
9	Google Cloud Storage Spark	add_spark_gcs()	name: str, bucket_or_name: str, gcs_options: dict
10	PostgreSQL	add_postgres()	name: str, connection_string: str
11	SQLite	add_sqlite()	name: str, connection_string: str
12	Snowflake	add_snowflake()	name: str, connection_string: str
13	DataBricks SQL	add_databricks_sql()	name: str, connection_string: str
14	Other SQL	add_sql()	name: str, connection_string: str

[1] https://docs.greatexpectations.io/docs/core/connect_to_data/

Adding a pandas Data Source

Use `.add_pandas()` to easily set up a Data Source for pandas DataFrames:

```
data_source = context.data_sources.add_pandas(
    name="my_pandas_datasource"
)
```

Retrieving components

Retrieve components with `.get()` by specifying their name parameter:

```
context.<COMPONENT>s.get(
    name: str
)
```

```
data_source = context.data_sources.get(
    name="my_pandas_datasource"
)
print(data_source)
```

```
id: 46c91f1b-1db9-4351-b5dd-83e038c0f511
name: 'my_pandas_datasource'
type: pandas
```

Retrieving components

Data Sources:

```
context.data_sources.get(
    name="my_pandas_datasource"
)
```

Validation Definitions:

```
context.validation_definitions.get(
    name="my_validation_definition"
)
```

Expectation Suites:

```
context.suites.get(
    name="my_suite"
)
```

Checkpoints:

```
context.checkpoints.get(
    name="my_checkpoint"
)
```

Listing components

Use `.all()` to list all components in your Data Context, including their names and metadata:

```
context.<COMPONENT>s.all()
```

```
data_sources = context.data_sources.all()
print(data_sources)
```

```
{
    'my_pandas_datasource': PandasDatasource(
        type='pandas',
        name='my_pandas_datasource',
        id=UUID('c22b16f7-6945-400e-932f-026cbd63b112'),
        assets=[]
    )
}
```

Listing components

Data Sources:

```
context.data_sources.all()
```

Validation Definitions:

```
context.validation_definitions.all()
```

Expectation Suites:

```
context.suites.all()
```

Checkpoints:

```
context.checkpoints.all()
```

Deleting components

Use `.delete()` to remove components by specifying their name:

```
context.<COMPONENT>s.delete(
    name: str
)
```

```
context.data_sources.delete(
    name="my_pandas_datasource"
)
print(context.data_sources.all())
```

```
{}
```

Deleting components

Data Sources:

```
context.data_sources.delete(
    name="my_pandas_datasource"
)
```

Validation Definitions:

```
context.validation_definitions.delete(
    name="my_validation_definition"
)
```

Expectation Suites:

```
context.suites.delete(
    name="my_suite"
)
```

Checkpoints:

```
context.checkpoints.delete(
    name="my_checkpoint"
)
```

Cheat sheet

Add a component to Data Context:

```
context.data_sources.add(data_source)

context.suites.add(suite)

context.validation_definitions.add(
    validation_definition
)

context.checkpoints.add(checkpoint)
```

Retrieve a component:

```
.get(name: str)
```

List components:

```
.all()
```

Delete a component:

```
.delete(name: str)
```

Let's practice!

INTRODUCTION TO DATA QUALITY WITH GREAT EXPECTATIONS