Package: sagemaker.workflow (via r-universe)

December 16, 2024

Type Package

Title sagemaker pipeline and workflows

Version 0.1.2.9000

Description `sagemaker` pipeline and workflows.

Imports fs, httr, lgr, jsonlite, methods, digest, processx, data.table, R6, sagemaker.core, sagemaker.common, sagemaker.mlcore, sagemaker.mlframework, urltools, uuid

Suggests crayon, testthat (>= 3.0.0), mockthat

Remotes DyfanJones/sagemaker-r-core, DyfanJones/sagemaker-r-common, DyfanJones/sagemaker-r-mlcore, DyfanJones/sagemaker-r-mlframework

License Apache License (>= 2.0)

Encoding UTF-8

RoxygenNote 7.1.2

Collate 'r_utils.R' 'workflow_utilities.R' 'workflow_functions.R' 'workflow_entities.R' 'workflow_retry.R' 'workflow_properties.R' 'workflow_steps.R' 'workflow__utils.R' 'workflow_airflow.R' 'workflow_callback_step.R' 'workflow_check_job_config.R' 'workflow_clarify_check_step.R' 'workflow_step_collections.R' 'workflow_parameters.R' 'workflow_execution_variables.R' 'workflow_conditions.R' 'workflow_condition_step.R' 'workflow_emr_step.R' 'workflow_fail_step.R' 'workflow_lambda_step.R' 'workflow_parallelism_config.R' 'workflow_pipeline_experiment_config.R' 'workflow_pipeline.R' 'workflow_quality_check_step.R' 'wrangler_ingestion.R' 'wrangler_processing.R' 'zzz.R'

Depends R (>= 2.10)

Config/testthat/edition 3

Config/pak/sysreqs make libxml2-dev libssl-dev

Repository https://dyfanjones.r-universe.dev

RemoteUrl https://github.com/DyfanJones/sagemaker-r-workflow

RemoteRef HEAD

RemoteSha 6f7c87c1993b6308066898fa2f0d2e871c4a711e

Contents

`sagemaker.workflow-package` *r6 sagemaker: this is just a placeholder*

Description

'sagemaker' pipeline and workflows.

Author(s)

Maintainer: Dyfan Jones <dyfan.r.jones@gmail.com>

Other contributors:

- Amazon.com, Inc. [copyright holder]

`CacheConfig` *Workflow CacheConfig class*

Description

Configuration class to enable caching in pipeline workflow.

Public fields

`enable_caching` To enable step caching.

`expire_after` If step caching is enabled, a timeout also needs to defined.

Active bindings

`config` Configures caching in pipeline steps.

Methods

Public methods:

- `CacheConfig$new()`
- `CacheConfig$format()`
- `CacheConfig$clone()`

Method `new()`**:** Initialize Workflow CacheConfig If caching is enabled, the pipeline attempts to find a previous execution of a step that was called with the same arguments. Step caching only considers successful execution. If a successful previous execution is found, the pipeline propagates the values from previous execution rather than recomputing the step. When multiple successful executions exist within the timeout period, it uses the result for the most recent successful execution.

Usage:

```
CacheConfig$new(enable_caching = FALSE, expire_after = NULL)
```

Arguments:

`enable_caching` (bool): To enable step caching. Defaults to 'FALSE'.

`expire_after` (str): If step caching is enabled, a timeout also needs to defined. It defines how old a previous execution can be to be considered for reuse. Value should be an ISO 8601 duration string. Defaults to 'NULL'.

Method `format()`**:** format class

Usage:

```
CacheConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CacheConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CallbackOutput` *Workflow CallbackOutput class*

Description

Output for a callback step.

Public fields

`output_name` The output name

`output_type` The output type

Methods

Public methods:

- `CallbackOutput$new()`
- `CallbackOutput$to_request()`
- `CallbackOutput$expr()`
- `CallbackOutput$format()`
- `CallbackOutput$clone()`

Method `new()`**:** Initialize CallbackOutput class

Usage:

```
CallbackOutput$new(output_name, output_type = CallbackOutputTypeEnum$String)
```

Arguments:

`output_name` (str): The output name

`output_type` (CallbackOutputTypeEnum): The output type

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
CallbackOutput$to_request()
```

Method `expr()`**:** The 'Get' expression dict for a 'CallbackOutput'.

Usage:

```
CallbackOutput$expr(step_name)
```

Arguments:

`step_name` (str): The name of the step the callback step associated with this output belongs to.

Method `format()`**:** format class

Usage:

```
CallbackOutput$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CallbackOutput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CallbackOutputTypeEnum`

Workflow CallbackOuputTypeEnum class

Description

CallbackOutput type enum.

Usage

```
CallbackOutputTypeEnum
```

Format

An object of class `Enum` (inherits from `environment`) of length 4.

CallbackStep *Callback step for workflow.*

Description

Callback step for workflow.

Callback step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `CallbackStep`

Public fields

`sqs_queue_url` An SQS queue URL for receiving callback messages.

`inputs` Input arguments that will be provided in the SQS message body of callback messages

`outputs` Outputs that can be provided when completing a callback.

`cache_config` A list of step names this 'TransformStep'

Active bindings

`arguments` The arguments dict that is used to define the callback step

`properties` A Properties object representing the output parameters of the callback step.

Methods

Public methods:

- `CallbackStep$new()`
- `CallbackStep$to_request()`
- `CallbackStep$clone()`

Method `new()`**:** Constructs a CallbackStep.

Usage:

```
CallbackStep$new(
  name,
  sqs_queue_url,
  inputs,
  outputs,
  display_name = NULL,
  description = NULL,
  cache_config = NULL,
  depends_on = NULL
)
```

Arguments:

`name` (str): The name of the callback step.

`sqs_queue_url` (str): An SQS queue URL for receiving callback messages.

`inputs` (dict): Input arguments that will be provided in the SQS message body of callback messages.

`outputs` (List[CallbackOutput]): Outputs that can be provided when completing a callback.

`display_name` (str): The display name of the callback step.

`description` (str): The description of the callback step.

`cache_config` (CacheConfig): A 'CacheConfig' instance.

`depends_on` (List[str]): A list of step names this 'TransformStep' depends on

Method `to_request()`**:** Updates the dictionary with cache configuration.

Usage:

```
CallbackStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CallbackStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CheckJobConfig` *CheckJobConfig class*

Description

Check job config for QualityCheckStep and ClarifyCheckStep

Methods

Public methods:

- `CheckJobConfig$new()`
- `CheckJobConfig$.generate_model_monitor()`
- `CheckJobConfig$format()`
- `CheckJobConfig$clone()`

Method `new()`**:** Constructs a CheckJobConfig instance.

Usage:

```
CheckJobConfig$new(
  role,
  instance_count = 1,
  instance_type = "ml.m5.xlarge",
  volume_size_in_gb = 30,
  volume_kms_key = NULL,
  output_kms_key = NULL,
```

```
  max_runtime_in_seconds = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  env = NULL,
  tags = NULL,
  network_config = NULL
)
```

Arguments:

`role` (str): An AWS IAM role. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with (default: 1).

`instance_type` (str): Type of EC2 instance to use for the job (default: 'ml.m5.xlarge').

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data during processing (default: 30).

`volume_kms_key` (str): A KMS key for the processing volume (default: None).

`output_kms_key` (str): The KMS key id for the job's outputs (default: None).

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600 if not specified

`base_job_name` (str): Prefix for the job name. If not specified, a default name is generated based on the training image name and current timestamp (default: None).

`sagemaker_session` (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed (default: None). If not specified, one is created using the default AWS configuration chain.

`env` (dict): Environment variables to be passed to the job (default: None).

`tags` ([dict]): List of tags to be passed to the job (default: None).

`network_config` (sagemaker.network.NetworkConfig): A NetworkConfig object that configures network isolation, encryption of inter-container traffic, security group IDs, and subnets (default: None).

Method `.generate_model_monitor()`**:** Generates a ModelMonitor object Generates a ModelMonitor object with required config attributes for QualityCheckStep and ClarifyCheckStep

Usage:

```
CheckJobConfig$.generate_model_monitor(mm_type)
```

Arguments:

`mm_type` (str): The subclass type of ModelMonitor object. A valid mm_type should be one of the following: "DefaultModelMonitor", "ModelQualityMonitor", "ModelBiasMonitor", "ModelExplainabilityMonitor"

Returns: sagemaker.model_monitor.ModelMonitor or None if the mm_type is not valid

Method `format()`**:** Format class

Usage:

```
CheckJobConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CheckJobConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ClarifyCheckConfig` *Clarify Check Config*

Description

Clarify Check Config

Clarify Check Config

Public fields

`data_config` Config of the input/output data.

`kms_key` The ARN of the KMS key that is used to encrypt the user code file

`monitoring_analysis_config_uri` The uri of monitoring analysis config.

Methods

Public methods:

- `ClarifyCheckConfig$new()`
- `ClarifyCheckConfig$format()`
- `ClarifyCheckConfig$clone()`

Method `new()`**:** Initialize ClarifyCheckConfig class

Usage:

```
ClarifyCheckConfig$new(
  data_config,
  kms_key = NULL,
  monitoring_analysis_config_uri = NULL
)
```

Arguments:

`data_config` (DataConfig): Config of the input/output data.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None). This field CANNOT be any of PipelineNonPrimitiveInputTypes.

`monitoring_analysis_config_uri` (str): The uri of monitoring analysis config. This field does not take input. It will be generated once uploading the created analysis config file.

Method `format()`**:** format class

Usage:

```
ClarifyCheckConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ClarifyCheckConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ClarifyCheckStep` *ClarifyCheckStep step for workflow.*

Description

ClarifyCheckStep step for workflow.

ClarifyCheckStep step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `ClarifyCheckStep`

Active bindings

`arguments` The arguments dict that is used to define the ClarifyCheck step.

`properties` A Properties object representing the output parameters of the ClarifyCheck step.

Methods

Public methods:

- `ClarifyCheckStep$new()`
- `ClarifyCheckStep$to_request()`
- `ClarifyCheckStep$clone()`

Method `new()`**:** Constructs a ClarifyCheckStep.

Usage:

```
ClarifyCheckStep$new(
  name,
  clarify_check_config,
  check_job_config,
  skip_check = FALSE,
  register_new_baseline = FALSE,
  model_package_group_name = NULL,
  supplied_baseline_constraints = NULL,
  display_name = NULL,
  description = NULL,
  cache_config = NULL,
  depends_on = NULL
)
```

Arguments:

`name` (str): The name of the ClarifyCheckStep step.

`clarify_check_config` (ClarifyCheckConfig): A ClarifyCheckConfig instance.

`check_job_config` (CheckJobConfig): A CheckJobConfig instance.

`skip_check` (bool or PipelineNonPrimitiveInputTypes): Whether the check should be skipped (default: False).

`register_new_baseline` (bool or PipelineNonPrimitiveInputTypes): Whether the new baseline should be registered (default: False).

`model_package_group_name` (str or PipelineNonPrimitiveInputTypes): The name of a registered model package group, among which the baseline will be fetched from the latest approved model (default: None).

`supplied_baseline_constraints` (str or PipelineNonPrimitiveInputTypes): The S3 path to the supplied constraints object representing the constraints JSON file which will be used for drift to check (default: None).

`display_name` (str): The display name of the ClarifyCheckStep step (default: None).

`description` (str): The description of the ClarifyCheckStep step (default: None).

`cache_config` (CacheConfig): A 'sagemaker.workflow.steps.CacheConfig' instance (default: None).

`depends_on` (List[str] or List[Step]): A list of step names or step instances this 'sagemaker.workflow.steps.ClarifyCheckS depends on (default: None).

Method `to_request()`**:** Updates the dictionary with cache configuration etc.

Usage:

```
ClarifyCheckStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ClarifyCheckStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CompilationStep` *CompilationStep class*

Description

Compilation step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `sagemaker.workflow::ConfigurableRetryStep` -> `CompilationStep`

Active bindings

`arguments` The arguments dict that is used to call 'create_compilation_job'. NOTE: The CreateTrainingJob request is not quite the args list that workflow needs. The TrainingJobName and ExperimentConfig attributes cannot be included.

`properties` A Properties object representing the DescribeTrainingJobResponse data model.

Methods

Public methods:

- `CompilationStep$new()`
- `CompilationStep$to_request()`
- `CompilationStep$clone()`

Method `new()`**:** Construct a CompilationStep. Given an 'EstimatorBase' and a 'sagemaker.model.Model' instance construct a CompilationStep. In addition to the estimator and Model instances, the other arguments are those that are supplied to the 'compile_model' method of the 'sagemaker.model.Model.compile_model'.

Usage:

```
CompilationStep$new(
  name,
  estimator,
  model,
  inputs = NULL,
  job_arguments = NULL,
  depends_on = NULL,
  retry_policies = NULL,
  display_name = NULL,
  description = NULL,
  cache_config = NULL
)
```

Arguments:

`name` (str): The name of the compilation step.

`estimator` (EstimatorBase): A 'sagemaker.estimator.EstimatorBase' instance.

`model` (Model): A 'sagemaker.model.Model' instance.

`inputs` (CompilationInput): A 'sagemaker.inputs.CompilationInput' instance. Defaults to 'None'.

`job_arguments` (List[str]): A list of strings to be passed into the processing job. Defaults to 'None'.

`depends_on` (List[str] or List[Step]): A list of step names or step instances this 'sagemaker.workflow.steps.CompilationSt depends on

`retry_policies` (List[RetryPolicy]): A list of retry policy

`display_name` (str): The display name of the compilation step.

`description` (str): The description of the compilation step.

`cache_config` (CacheConfig): A 'sagemaker.workflow.steps.CacheConfig' instance.

Method `to_request()`**:** Updates the dictionary with cache configuration.

Usage:

```
CompilationStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CompilationStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Condition` *Workflow Condition class*

Description

Abstract Condition entity.

Super class

`sagemaker.workflow::Entity` -> `Condition`

Public fields

`condition_type` The type of condition.

Methods

Public methods:

- `Condition$new()`
- `Condition$clone()`

Method `new()`**:** Initialize Condition class

Usage:

```
Condition$new(condition_type = enum_items(ConditionTypeEnum))
```

Arguments:

`condition_type` (ConditionTypeEnum): The type of condition.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Condition$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionComparison *Workflow ConditionComparison class*

Description

Generic comparison condition that can be used to derive specific condition comparisons.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Condition -> ConditionComparison

Public fields

`left` The execution variable, parameter, or property to use in the comparison.

`right` The execution variable, parameter, property, or Python primitive value to compare to.

Methods

Public methods:

- ConditionComparison$new()
- ConditionComparison$to_request()
- ConditionComparison$clone()

Method `new()`**:** Initialize ConditionComparison Class

Usage:

```
ConditionComparison$new(
  condition_type = enum_items(ConditionTypeEnum),
  left,
  right
)
```

Arguments:

`condition_type` (ConditionTypeEnum): The type of condition.

`left` (ConditionValueType): The execution variable, parameter, or property to use in the comparison.

`right` (Union[ConditionValueType, PrimitiveType]): The execution variable, parameter, property, or Python primitive value to compare to.

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
ConditionComparison$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionComparison$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionEquals *Workflow ConditionEquals Class*

Description

A condition for equality comparisons.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Condition -> sagemaker.workflow::ConditionComparison -> ConditionEquals

Methods

Public methods:

- ConditionEquals$new()
- ConditionEquals$clone()

Method `new()`**:** Construct A condition for equality comparisons.

Usage:

```
ConditionEquals$new(left, right)
```

Arguments:

`left` (ConditionValueType): The execution variable, parameter, or property to use in the comparison.

`right` (Union[ConditionValueType, PrimitiveType]): The execution variable, parameter, property, or Python primitive value to compare to.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionEquals$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionGreaterThan *Workflow ConditionGreaterThan Class*

Description

A condition for greater than comparisons.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Condition -> sagemaker.workflow::ConditionComparison -> ConditionGreaterThan

Methods

Public methods:

- ConditionGreaterThan$new()
- ConditionGreaterThan$clone()

Method `new()`**:** Construct an instance of ConditionGreaterThan for greater than comparisons.

Usage:

```
ConditionGreaterThan$new(left, right)
```

Arguments:

`left` (ConditionValueType): The execution variable, parameter, or property to use in the comparison.

`right` (Union[ConditionValueType, PrimitiveType]): The execution variable, parameter, property, or Python primitive value to compare to.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionGreaterThan$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionGreaterThanOrEqualTo

Workflow ConditionGreaterThanOrEqualTo class

Description

A condition for greater than or equal to comparisons.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Condition` -> `sagemaker.workflow::ConditionComparison` -> `ConditionGreaterThanOrEqualTo`

Methods

Public methods:

- ConditionGreaterThanOrEqualTo$new()
- ConditionGreaterThanOrEqualTo$clone()

Method `new()`**:** Construct of ConditionGreaterThanOrEqualTo for greater than or equal to comparisons.

Usage:

```
ConditionGreaterThanOrEqualTo$new(left, right)
```

Arguments:

`left` (ConditionValueType): The execution variable, parameter, or property to use in the comparison.

`right` (Union[ConditionValueType, PrimitiveType]): The execution variable, parameter, property, or Python primitive value to compare to.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionGreaterThanOrEqualTo$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ConditionIn` *Workflow ConditionIn class*

Description

A condition to check membership.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Condition` -> `ConditionIn`

Methods

Public methods:

- `ConditionIn$new()`
- `ConditionIn$to_request()`
- `ConditionIn$clone()`

Method `new()`**:** Construct a 'ConditionIn' condition to check membership.

Usage:

```
ConditionIn$new(value, in_values)
```

Arguments:

`value` (ConditionValueType): The execution variable, parameter, or property to use for the in comparison.

`in_values` (List[Union[ConditionValueType, PrimitiveType]]): The list of values to check for membership in.

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
ConditionIn$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionIn$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ConditionLessThan` *Workflow ConditionLessThan class*

Description

A condition for less than or equal to comparisons.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Condition` -> `sagemaker.workflow::ConditionComparison` -> `ConditionLessThan`

Methods

Public methods:

- `ConditionLessThan$new()`
- `ConditionLessThan$clone()`

Method `new()`**:** Construct ConditionLessThanOrEqualTo for less than or equal to comparisons.

Usage:

```
ConditionLessThan$new(left, right)
```

Arguments:

`left` (ConditionValueType): The execution variable, parameter, or property to use in the comparison.

`right` (Union[ConditionValueType, PrimitiveType]): The execution variable, parameter, property, or Python primitive value to compare to.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionLessThan$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionLessThanOrEqualTo *Workflow ConditionLessThanOrEqualTo class*

Description

A condition for less than or equal to comparisons.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Condition -> sagemaker.workflow::ConditionComparison -> ConditionLessThanOrEqualTo

Methods

Public methods:

- ConditionLessThanOrEqualTo$new()
- ConditionLessThanOrEqualTo$clone()

Method `new()`**:** Construct ConditionLessThanOrEqualTo for less than or equal to comparisons.

Usage:

```
ConditionLessThanOrEqualTo$new(left, right)
```

Arguments:

`left` (ConditionValueType): The execution variable, parameter, or property to use in the comparison.

`right` (Union[ConditionValueType, PrimitiveType]): The execution variable, parameter, property, or Python primitive value to compare to.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionLessThanOrEqualTo$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionNot *Workflow ConditionNot class*

Description

A condition for negating another 'Condition'.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Condition` -> `ConditionNot`

Methods

Public methods:

- `ConditionNot$new()`
- `ConditionNot$to_request()`
- `ConditionNot$clone()`

Method `new()`**:** Construct a 'ConditionNot' condition for negating another 'Condition'.

Usage:

```
ConditionNot$new(expression)
```

Arguments:

`expression` (Condition): A 'Condition' to take the negation of.

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
ConditionNot$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionNot$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionOr *Workflow ConditionOr class*

Description

A condition for taking the logical OR of a list of 'Condition' instances.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Condition` -> `ConditionOr`

Methods

Public methods:

- `ConditionOr$new()`
- `ConditionOr$to_request()`
- `ConditionOr$clone()`

Method `new()`**:** Construct a 'ConditionOr' condition.

Usage:

```
ConditionOr$new(conditions = NULL)
```

Arguments:

`conditions` (List[Condition]): A list of 'Condition' instances to logically OR.

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
ConditionOr$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionOr$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ConditionStep *Workflow ConditionStep class*

Description

Conditional step for pipelines to support conditional branching in the execution of steps.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `ConditionStep`

Public fields

`conditions` The name of the step.

`if_steps` A list of 'sagemaker.workflow.steps.Step' and 'sagemaker.workflow.step_collections.StepCollection' instances

`else_steps` A list of 'sagemaker.workflow.steps.Step' and 'sagemaker.workflow.step_collections.StepCollection' instances

Active bindings

`arguments` The arguments dict that is used to define the conditional branching in the pipeline.

`properties` A simple Properties object with 'Outcome' as the only property

Methods

Public methods:

- `ConditionStep$new()`
- `ConditionStep$clone()`

Method `new()`**:** Construct a ConditionStep for pipelines to support conditional branching. If all of the conditions in the condition list evaluate to True, the 'if_steps' are marked as ready for execution. Otherwise, the 'else_steps' are marked as ready for execution.

Usage:

```
ConditionStep$new(
  name,
  depends_on = NULL,
  display_name = NULL,
  description = NULL,
  conditions = NULL,
  if_steps = NULL,
  else_steps = NULL
)
```

Arguments:

`name` (str): The name of the step.

depends_on (List[str]): The list of step names the current step depends on

display_name (str): The display name of the condition step.

description (str): The description of the condition step.

conditions (List[Condition]): A list of ‘sagemaker.workflow.conditions.Condition‘ instances.

if_steps (List[Union[Step, StepCollection]]): A list of ‘sagemaker.workflow.steps.Step‘ and ‘sagemaker.workflow.step_collections.StepCollection‘ instances that are marked as ready for execution if the list of conditions evaluates to True.

else_steps (List[Union[Step, StepCollection]]): A list of ‘sagemaker.workflow.steps.Step‘ and ‘sagemaker.workflow.step_collections.StepCollection‘ instances that are marked as ready for execution if the list of conditions evaluates to False.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConditionStep$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`ConditionTypeEnum` *Workflow CondiitionType class*

Description

Condition type enum.

Usage

```
ConditionTypeEnum
```

Format

An object of class `Enum` (inherits from `environment`) of length 8.

`ConfigurableRetryStep` *ConfigurableRetryStep class*

Description

ConfigurableRetryStep step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `ConfigurableRetryStep`

Methods

Public methods:

- ConfigurableRetryStep$new()
- ConfigurableRetryStep$add_retry_policy()
- ConfigurableRetryStep$to_request()
- ConfigurableRetryStep$clone()

Method `new()`**:** Initialize ConfigurableRetryStep class

Usage:

```
ConfigurableRetryStep$new(
  name,
  step_type = enum_items(StepTypeEnum),
  display_name = NULL,
  description = NULL,
  depends_on = NULL,
  retry_policies = NULL
)
```

Arguments:

`name` (str): The name of the step.

`step_type` (StepTypeEnum): The type of the step.

`display_name` (str): The display name of the step.

`description` (str): The description of the step.

`depends_on` (List[str] or List[Step]): The list of step names or step instances the current step depends on

`retry_policies` (List[RetryPolicy]): The custom retry policy configuration

Method `add_retry_policy()`**:** Add a retry policy to the current step retry policies list.

Usage:

```
ConfigurableRetryStep$add_retry_policy(retry_policy)
```

Arguments:

`retry_policy` : Placeholder

Method `to_request()`**:** Gets the request structure for ConfigurableRetryStep

Usage:

```
ConfigurableRetryStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ConfigurableRetryStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

CreateModelStep *Workflow CreateModel class*

Description

CreateModel step for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Step -> sagemaker.workflow::ConfigurableRetryStep -> CreateModelStep

Active bindings

`arguments` The arguments dict that is used to call 'create_model'. NOTE: The CreateModelRequest is not quite the args list that workflow needs. ModelName cannot be included in the arguments.

`properties` A Properties object representing the DescribeModelResponse data model.

Methods

Public methods:

- `CreateModelStep$new()`
- `CreateModelStep$clone()`

Method `new()`**:** Construct a CreateModelStep, given an 'sagemaker.model.Model' instance. In addition to the Model instance, the other arguments are those that are supplied to the '_create_sagemaker_model' method of the 'sagemaker.model.Model._create_sagemaker_model'.

Usage:

```
CreateModelStep$new(
  name,
  model,
  inputs,
  depends_on = NULL,
  retry_policies = NULL,
  display_name = NULL,
  description = NULL
)
```

Arguments:

`name` (str): The name of the CreateModel step.

`model` (Model): A 'sagemaker.model.Model' instance.

`inputs` (CreateModelInput): A 'sagemaker.inputs.CreateModelInput' instance. Defaults to 'None'.

`depends_on` (List[str]): A list of step names this 'sagemaker.workflow.steps.CreateModelStep' depends on

retry_policies (List[RetryPolicy]): A list of retry policy
display_name (str): The display name of the CreateModel step.
description (str): The description of the CreateModel step.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CreateModelStep$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`DataBiasCheckConfig` *Data Bias Check Config*

Description

Data Bias Check Config

Data Bias Check Config

Super class

`sagemaker.workflow::ClarifyCheckConfig` -> `DataBiasCheckConfig`

Public fields

`data_bias_config` Config of sensitive groups

`methods` Selector of a subset of potential metrics

Methods

Public methods:

- `DataBiasCheckConfig$new()`
- `DataBiasCheckConfig$clone()`

Method `new()`**:** Initialize DataBiasCheckConfig class

Usage:

```
DataBiasCheckConfig$new(data_bias_config, methods = "all", ...)
```

Arguments:

`data_bias_config` (BiasConfig): Config of sensitive groups.

`methods` (str or list[str]): Selector of a subset of potential metrics:

- "CI" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-bias-metric-class-imbalance.html,
- "DPL" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-true-label-i

- "KL" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-kl-divergence
html,
- "JS" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-jensen-shannon
html,
- "LP" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-lp-norm.
html,
- "TVD" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-total-varia
html,
- "KS" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-kolmogorov-sm
html,
- "CDDL" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-data-bias-metric-cddl.
html

Defaults to computing all. This field CANNOT be any of PipelineNonPrimitiveInputTypes.

... : Parameters from ClarifyCheckConfig

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataBiasCheckConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DataQualityCheckConfig`

DataQualityCheckConfig class

Description

Data Quality Check Config.

Super class

`sagemaker.workflow::QualityCheckConfig` -> `DataQualityCheckConfig`

Public fields

`record_preprocessor_script` (str): The path to the record preprocessor script (default: None). This can be a local path or an S3 uri string but CANNOT be any of PipelineNonPrimitiveInputTypes.

Methods

Public methods:

- `DataQualityCheckConfig$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataQualityCheckConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DataWranglerProcessor` *DataWranglerProcessor class*

Description

Handles Amazon SageMaker DataWrangler tasks

Super class

`sagemaker.common::Processor`

Methods

Public methods:

- `DataWranglerProcessor$new()`
- `DataWranglerProcessor$clone()`

Method `new()`**:** Initializes a "Processor" instance. The "Processor" handles Amazon SageMaker Processing tasks.

Usage:

```
DataWranglerProcessor$new(
  role,
  data_wrangler_flow_source,
  instance_count,
  instance_type,
  volume_size_in_gb = 30L,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  env = NULL,
  tags = NULL,
  network_config = NULL
)
```

Arguments:

`role` (str): An AWS IAM role name or ARN. Amazon SageMaker Processing uses this role to access AWS resources, such as data stored in Amazon S3.

`data_wrangler_flow_source` (str): The source of the DaraWrangler flow which will be used for the DataWrangler job. If a local path is provided, it will automatically be uploaded to S3 under: "s3://<default-bucket-name>/<job-name>/input/<input-name>".

`instance_count` (int): The number of instances to run a processing job with.

`instance_type` (str): The type of EC2 instance to use for processing, for example, 'ml.c4.xlarge'.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data during processing (default: 30).

`volume_kms_key` (str): A KMS key for the processing volume (default: None).

`output_kms_key` (str): The KMS key ID for processing job outputs (default: None).

`max_runtime_in_seconds` (int): Timeout in seconds (default: None). After this amount of time, Amazon SageMaker terminates the job, regardless of its current status. If 'max_runtime_in_seconds' is not specified, the default value is 24 hours.

`base_job_name` (str): Prefix for processing job name. If not specified, the processor generates a default job name, based on the processing image name and current timestamp.

`sagemaker_session` (:class:'~sagemaker.session.Session'): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

`env` (dict[str, str]): Environment variables to be passed to the processing jobs (default: None).

`tags` (list[dict]): List of tags to be passed to the processing job (default: None). For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

`network_config` (:class:'~sagemaker.network.NetworkConfig'): A :class:'~sagemaker.network.NetworkConfig' object that configures network isolation, encryption of inter-container traffic, security group IDs, and subnets.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataWranglerProcessor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`deploy_config` *Export Airflow deploy config from a SageMaker model*

Description

Export Airflow deploy config from a SageMaker model

Usage

```
deploy_config(
  model,
  initial_instance_count,
  instance_type,
  endpoint_name = NULL,
  tags = NULL
)
```

Arguments

model	(sagemaker.model.Model): The SageMaker model to export the Airflow config from.
initial_instance_count	(int): The initial number of instances to run in the "Endpoint" created from this "Model".
instance_type	(str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.
endpoint_name	(str): The name of the endpoint to create (default: None). If not specified, a unique endpoint name will be created.
tags	(list[dict]): List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/

Value

dict: Deploy config that can be directly used by SageMakerEndpointOperator in Airflow.

---

deploy_config_from_estimator *Export Airflow deploy config from a SageMaker estimator*

Description

Export Airflow deploy config from a SageMaker estimator

Usage

```
deploy_config_from_estimator(
  estimator,
  task_id,
  task_type,
  initial_instance_count,
  instance_type,
  model_name = NULL,
  endpoint_name = NULL,
  tags = NULL,
  ...
)
```

Arguments

estimator	(sagemaker.model.EstimatorBase): The SageMaker estimator to export Airflow config from. It has to be an estimator associated with a training job.
task_id	(str): The task id of any airflow.contrib.operators.SageMakerTrainingOperator or airflow.contrib.operators.SageMakerTuningOperator that generates training jobs in the DAG. The endpoint config is built based on the training job generated in this operator.

`task_type` (str): Whether the task is from SageMakerTrainingOperator or SageMakerTuningOperator. Values can be 'training', 'tuning' or None (which means training job is not from any task).

`initial_instance_count` (int): Minimum number of EC2 instances to deploy to an endpoint for prediction.

`instance_type` (str): Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.

`model_name` (str): Name to use for creating an Amazon SageMaker model. If not specified, one will be generated.

`endpoint_name` (str): Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the SageMaker model is used.

`tags` (list[dict]): List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/

`...` : Passed to invocation of "create_model()". Implementations may customize "create_model()" to accept "**kwargs" to customize model creation during deploy. For more, see the implementation docs.

Value

dict: Deploy config that can be directly used by SageMakerEndpointOperator in Airflow.

`EMRStep` *EMRStep class*

Description

EMR step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `EMRStep`

Active bindings

`arguments` The arguments dict that is used to call 'AddJobFlowSteps'. NOTE: The AddFlowJobSteps request is not quite the args list that workflow needs. The Name attribute in AddJobFlowSteps cannot be passed; it will be set during runtime. In addition to that, we will also need to include emr job inputs and output config.

`properties` A Properties object representing the EMR DescribeStepResponse model

Methods

Public methods:

- EMRStep$new()
- EMRStep$to_request()
- EMRStep$clone()

Method `new()`**:** Constructs a EMRStep.

Usage:

```
EMRStep$new(
  name,
  display_name,
  description,
  cluster_id,
  step_config,
  depends_on = NULL,
  cache_config = NULL
)
```

Arguments:

`name` (str): The name of the EMR step.

`display_name` (str): The display name of the EMR step.

`description` (str): The description of the EMR step.

`cluster_id` (str): The ID of the running EMR cluster.

`step_config` (EMRStepConfig): One StepConfig to be executed by the job flow.

`depends_on` (List[str]): A list of step names this 'sagemaker.workflow.steps.EMRStep' depends on

`cache_config` (CacheConfig): A 'sagemaker.workflow.steps.CacheConfig' instance.

Method `to_request()`**:** Updates the dictionary with cache configuration.

Usage:

```
EMRStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
EMRStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

EMRStepConfig *EMRStepConfig class*

Description

Config for a Hadoop Jar step

Public fields

`jar` A path to a JAR file run during the step.

`args` A list of command line arguments

`main_class` The name of the main class in the specified Java file.

`properties` A list of key-value pairs that are set when the step runs.

Methods

Public methods:

- `EMRStepConfig$new()`
- `EMRStepConfig$to_request()`
- `EMRStepConfig$format()`
- `EMRStepConfig$clone()`

Method `new()`**:** Create a definition for input data used by an EMR cluster(job flow) step. See AWS documentation on the "StepConfig" API for more details on the parameters.

Usage:

```
EMRStepConfig$new(jar, args = NULL, main_class = NULL, properties = NULL)
```

Arguments:

`jar` (str): A path to a JAR file run during the step.

`args` (List[str]): A list of command line arguments passed to the JAR file's main function when executed.

`main_class` (str): The name of the main class in the specified Java file.

`properties` (List(dict)): A list of key-value pairs that are set when the step runs.

Method `to_request()`**:** Convert EMRStepConfig object to request list.

Usage:

```
EMRStepConfig$to_request()
```

Method `format()`**:** format class

Usage:

```
EMRStepConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
EMRStepConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`EstimatorTransformer` *Workflow EstimatorTransformer class*

Description

Creates a Transformer step collection for workflow.

Super class

`sagemaker.workflow::StepCollection` -> `EstimatorTransformer`

Methods

Public methods:

- `EstimatorTransformer$new()`
- `EstimatorTransformer$clone()`

Method `new()`**:** Construct steps required for a Transformer step collection: An estimator-centric step collection. It models what happens in workflows when invoking the 'transform()' method on an estimator instance: First, if custom model artifacts are required, a '_RepackModelStep' is included. Second, a 'CreateModelStep' with the model data passed in from a training step or other training job output. Finally, a 'TransformerStep'. If repacking the model artifacts is not necessary, only the CreateModelStep and TransformerStep are in the step collection.

Usage:

```
EstimatorTransformer$new(
  name,
  estimator,
  model_data,
  model_inputs,
  instance_count,
  instance_type,
  transform_inputs,
  description = NULL,
  display_name = NULL,
  image_uri = NULL,
  predictor_cls = NULL,
  env = NULL,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  volume_kms_key = NULL,
```

```
  depends_on = NULL,
  repack_model_step_retry_policies = NULL,
  model_step_retry_policies = NULL,
  transform_step_retry_policies = NULL,
  ...
)
```

Arguments:

`name` (str): The name of the Transform Step.

`estimator` : The estimator instance.

`model_data` (str): The S3 location of a SageMaker model data ".tar.gz" file (default: None).

`model_inputs` (CreateModelInput): A 'sagemaker.inputs.CreateModelInput' instance. Defaults to 'None'.

`instance_count` (int): The number of EC2 instances to use.

`instance_type` (str): The type of EC2 instance to use.

`transform_inputs` (TransformInput): A 'sagemaker.inputs.TransformInput' instance.

`description` (str): The description of the step.

`display_name` (str): The display name of the step.

`image_uri` (str): A Docker image URI.

`predictor_cls` (callable[string, :Session]): A function to call to create a predictor (default: None). If not None, "deploy" will return the result of invoking this function on the created endpoint name.

`env` (dict): The Environment variables to be set for use during the transform job (default: None).

`strategy` (str): The strategy used to decide how to batch records in a single request (default: None). Valid values: 'MultiRecord' and 'SingleRecord'.

`assemble_with` (str): How the output is assembled (default: None). Valid values: 'Line' or 'None'.

`output_path` (str): The S3 location for saving the transform result. If not specified, results are stored to a default bucket.

`output_kms_key` (str): Optional. A KMS key ID for encrypting the transform output (default: None).

`accept` (str): The accept header passed by the client to the inference endpoint. If it is supported by the endpoint, it will be the format of the batch transform output.

`max_concurrent_transforms` (int): The maximum number of HTTP requests to be made to each individual transform container at one time.

`max_payload` (int): Maximum size of the payload in a single HTTP

`tags` (list[dict]): List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

`depends_on` (List[str] or List[Step]): The list of step names or step instances the first step in the collection depends on

`repack_model_step_retry_policies` (List[RetryPolicy]): The list of retry policies for the repack model step

`model_step_retry_policies` (List[RetryPolicy]): The list of retry policies for model step

`transform_step_retry_policies` (List[RetryPolicy]): The list of retry policies for transform step

`...` : pass onto model class.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
EstimatorTransformer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ExecutionVariable` *Workflow ExecutionVariable class*

Description

Pipeline execution variables for workflow.

Super class

`sagemaker.workflow::Expression` -> `ExecutionVariable`

Public fields

`name` The name of the execution variable.

Active bindings

`expr` The 'Get' expression dict for an 'ExecutionVariable'.

Methods

Public methods:

- `ExecutionVariable$new()`
- `ExecutionVariable$clone()`

Method `new()`**:** Create a pipeline execution variable.

Usage:

```
ExecutionVariable$new(name)
```

Arguments:

`name` (str): The name of the execution variable.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ExecutionVariable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ExecutionVariables *Enum-like class for all ExecutionVariable instances.*

Description

Considerations to move these as module-level constants should be made.

Usage

```
ExecutionVariables
```

Format

An object of class `Enum` (inherits from `environment`) of length 6.

FailStep *Workflow FailStep*

Description

FailStep‘ for SageMaker Pipelines Workflows.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `FailStep`

Public fields

`error_message` An error message defined by the user.

Active bindings

`arguments` The arguments dictionary that is used to define the ‘FailStep‘.

`properties` A ‘Properties‘ object is not available for the ‘FailStep‘. Executing a ‘FailStep‘ will terminate the pipeline. ‘FailStep‘ properties should not be referenced.

Methods

Public methods:

- `FailStep$new()`
- `FailStep$clone()`

Method `new()`**:** Constructs a ‘FailStep‘.

Usage:

```
FailStep$new(
  name,
  error_message = NULL,
  display_name = NULL,
  description = NULL,
  depends_on = NULL
)
```

Arguments:

`name` (str): The name of the 'FailStep'. A name is required and must be unique within a pipeline.

`error_message` (str or PipelineNonPrimitiveInputTypes): An error message defined by the user. Once the 'FailStep' is reached, the execution fails and the error message is set as the failure reason (default: None).

`display_name` (str): The display name of the 'FailStep'. The display name provides better UI readability. (default: None).

`description` (str): The description of the 'FailStep' (default: None).

`depends_on` (List[str] or List[Step]): A list of 'Step' names or 'Step' instances that this 'FailStep' depends on. If a listed 'Step' name does not exist, an error is returned (default: None).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FailStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`format_start_parameters` *Formats start parameter overrides as a list of dicts.*

Description

This list of dicts adheres to the request schema of: '"Name": "MyParameterName", "Value": "MyValue"'

Usage

```
format_start_parameters(parameters)
```

Arguments

`parameters`	(Dict[str, Any]): A dict of named values where the keys are the names of the parameters to pass values into.

`generate_data_ingestion_flow_from_athena_dataset_definition`
generate data ingestion flow from athena dataset definition

Description

Generate the data ingestion only flow from athena input

Usage

```
generate_data_ingestion_flow_from_athena_dataset_definition(
  input_name,
  athena_dataset_definition,
  operator_version = "0.1",
  schema = NULL
)
```

Arguments

`input_name`	(str): the name of the input to flow source node
`athena_dataset_definition`	(AthenaDatasetDefinition): athena input to flow source node
`operator_version`	(str): the version of the operator
`schema`	(list): the schema for the data to be ingested

Value

dict (typing.Dict): A flow only conduct data ingestion with 1-1 mapping output_name (str): The output name used to configure ‘sagemaker.processing.FeatureStoreOutput‘

`generate_data_ingestion_flow_from_redshift_dataset_definition`
generate data ingestion flow from redshift dataset definition

---

Description

Generate the data ingestion only flow from redshift input

Usage

```
generate_data_ingestion_flow_from_redshift_dataset_definition(
  input_name,
  redshift_dataset_definition,
  operator_version = "0.1",
  schema = NULL
)
```

Arguments

input_name	(str): the name of the input to flow source node
redshift_dataset_definition	(RedshiftDatasetDefinition): redshift input to flow source node
operator_version	(str): the version of the operator
schema	(list): the schema for the data to be ingested

Value

list: A flow only conduct data ingestion with 1-1 mapping output_name (str): The output name used to configure ‘sagemaker.processing.FeatureStoreOutput‘

---

generate_data_ingestion_flow_from_s3_input

generate data ingestion flow from s3 input

Description

Generate the data ingestion only flow from s3 input

Usage

```
generate_data_ingestion_flow_from_s3_input(
  input_name,
  s3_uri,
  s3_content_type = "csv",
  s3_has_header = FALSE,
  operator_version = "0.1",
  schema = NULL
)
```

Arguments

input_name	(str): the name of the input to flow source node
s3_uri	(str): uri for the s3 input to flow source node
s3_content_type	(str): s3 input content type
s3_has_header	(bool): flag indicating the input has header or not
operator_version	(str): the version of the operator
schema	(list): the schema for the data to be ingested

Value

list: A flow only conduct data ingestion with 1-1 mapping output_name (str): The output name used to configure ‘sagemaker.processing.FeatureStoreOutput‘

`hash_file` *Get the MD5 hash of a file.*

Description

Get the MD5 hash of a file.

Usage

```
hash_file(path)
```

Arguments

`path`	(str): The local path for the file.

Value

str: The MD5 hash of the file.

`input_output_list_converter` *Converts a list of ProcessingInput or ProcessingOutput objects to a list of dicts*

Description

Converts a list of ProcessingInput or ProcessingOutput objects to a list of dicts

Usage

```
input_output_list_converter(object_list)
```

Arguments

`object_list`	(list[ProcessingInput or ProcessingOutput]

Value

List of dicts

`interpolate` *Replaces Parameter values in a list of nested Dict[str, Any] with their workflow expression.*

Description

Replaces Parameter values in a list of nested Dict[str, Any] with their workflow expression.

Usage

```
interpolate(
  request_obj,
  callback_output_to_step_map,
  lambda_output_to_step_map
)
```

Arguments

`request_obj`	(RequestType): The request dict.
`callback_output_to_step_map`	(list[str, str]): A dict of output name -> step name.
`lambda_output_to_step_map`	(list[str, str]): Placeholder

Value

RequestType: The request dict with Parameter values replaced by their expression.

`Join` *Join Class*

Description

Join together properties.

Super class

`sagemaker.workflow::Expression` -> `Join`

Public fields

`on` The primitive types and parameters to join.

`values` The string to join the values on (Defaults to "").

Active bindings

`expr` The expression dict for a 'Join' function.

Methods

Public methods:

- `Join$new()`
- `Join$clone()`

Method `new()`**:** Initialize Join Class

Usage:

```
Join$new(on = "", values = "")
```

Arguments:

`on` (str): The string to join the values on (Defaults to "").

`values` (List[Union[PrimitiveType, Parameter]]): The primitive types and parameters to join.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Join$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`JsonGet` *Workflow JsonGet class*

Description

Get JSON properties from PropertyFiles.

Super class

`sagemaker.workflow::Expression` -> `JsonGet`

Public fields

`step_name` The step from which to get the property file.

`property_file` Either a PropertyFile instance or the name of a property file.

`json_path` The JSON path expression to the requested value.

Active bindings

`expr` The expression dict for a 'JsonGet' function.

Methods

Public methods:

- JsonGet$new()
- JsonGet$clone()

Method `new()`**:** Initialize JsonGet class

Usage:

```
JsonGet$new(step_name, property_file, json_path)
```

Arguments:

`step_name` (Step): The step from which to get the property file.

`property_file` (Union[PropertyFile, str]): Either a PropertyFile instance or the name of a property file.

`json_path` (str): The JSON path expression to the requested value.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
JsonGet$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LambdaOutput` *Workflow LambdaOutput class*

Description

Output for a lambdaback step.

Public fields

`output_name` (str): The output name

`output_type` (LambdaOutputTypeEnum): The output type

Methods

Public methods:

- LambdaOutput$new()
- LambdaOutput$to_request()
- LambdaOutput$expr()
- LambdaOutput$format()
- LambdaOutput$clone()

Method `new()`**:** Initialize LambdaOutput class

Usage:

```
LambdaOutput$new(output_name, output_type = enum_items(LambdaOutputTypeEnum))
```

Arguments:

`output_name` (str): The output name

`output_type` (LambdaOutputTypeEnum): The output type

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
LambdaOutput$to_request()
```

Method `expr()`**:** The 'Get' expression dict for a 'LambdaOutput'.

Usage:

```
LambdaOutput$expr(step_name)
```

Arguments:

`step_name` (str): The name of the step the lambda step associated

Method `format()`**:** format class

Usage:

```
LambdaOutput$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LambdaOutput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LambdaOutputTypeEnum` *LambdaOutput type enum.*

Description

LambdaOutput type enum.

Usage

```
LambdaOutputTypeEnum
```

Format

An object of class `Enum` (inherits from `environment`) of length 4.

LambdaStep *Workflow LambdaStep class*

Description

Lambda step for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Step -> LambdaStep

Active bindings

`arguments` The arguments dict that is used to define the lambda step.

`properties` A Properties object representing the output parameters of the lambda step.

Methods

Public methods:

- LambdaStep$new()
- LambdaStep$to_request()
- LambdaStep$clone()

Method `new()`**:** Constructs a LambdaStep.

Usage:

```
LambdaStep$new(
  name,
  lambda_func,
  display_name = NULL,
  description = NULL,
  inputs = NULL,
  outputs = NULL,
  cache_config = NULL,
  depends_on = NULL
)
```

Arguments:

`name` (str): The name of the lambda step.

`lambda_func` (str): An instance of sagemaker.lambda_helper.Lambda. If lambda arn is specified in the instance, LambdaStep just invokes the function, else lambda function will be created while creating the pipeline.

`display_name` (str): The display name of the Lambda step.

`description` (str): The description of the Lambda step.

`inputs` (dict): Input arguments that will be provided to the lambda function.

`outputs` (List[LambdaOutput]): List of outputs from the lambda function.

cache_config (CacheConfig): A ‘sagemaker.workflow.steps.CacheConfig‘ instance.

depends_on (List[str]): A list of step names this ‘sagemaker.workflow.steps.LambdaStep‘ depends on

Method `to_request()`**:** Updates the dictionary with cache configuration.

Usage:

```
LambdaStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LambdaStep$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`list_to_request` *Get the request structure for list of entities.*

Description

Get the request structure for list of entities.

Usage

```
list_to_request(entities)
```

Arguments

entities	(Sequence[Entity]): A list of entities.

Value

list: A request structure for a workflow service call.

ModelBiasCheckConfig *Model Bias Check Config*

Description

Model Bias Check Config

Model Bias Check Config

Super class

sagemaker.workflow::ClarifyCheckConfig -> ModelBiasCheckConfig

Public fields

`data_bias_config` Config of sensitive groups

`model_config` Config of the model and its endpoint to be created

`model_predicted_label_config` Config of how to extract the predicted label from the model output

`methods` Selector of a subset of potential metrics

Methods

Public methods:

- ModelBiasCheckConfig$new()
- ModelBiasCheckConfig$clone()

Method `new()`**:** Initialize DataBiasCheckConfig class

Usage:

```
ModelBiasCheckConfig$new(
  data_bias_config,
  model_config,
  model_predicted_label_config,
  methods = "all",
  ...
)
```

Arguments:

`data_bias_config` (BiasConfig): Config of sensitive groups.

`model_config` (ModelConfig): Config of the model and its endpoint to be created.

`model_predicted_label_config` (ModelPredictedLabelConfig): Config of how to extract the predicted label from the model output.

`methods` (str or list[str]): Selector of a subset of potential metrics:

- "DPPL"https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dp
  html,

- "DI" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-di.html,
- "DCA" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dc html,
- "DCR" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dc html,
- "RD" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-rd.html,
- "DAR" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-da html,
- "DRR" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dr html,
- "AD" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ad.html,
- "CDDPL" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric- html,
- "TE" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-te.html,
- "FT" https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ft.html

Defaults to computing all. This field CANNOT be any of PipelineNonPrimitiveInputTypes.

... : Parameters from ClarifyCheckConfig

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ModelBiasCheckConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelExplainabilityCheckConfig`

Model Explainability Check Config

Description

Model Explainability Check Config

Model Explainability Check Config

Super class

`sagemaker.workflow::ClarifyCheckConfig` -> `ModelExplainabilityCheckConfig`

Public fields

`model_config` Config of the model and its endpoint to be created

`explainability_config` Config of the specific explainability method

`model_scores` Index or JSONPath location in the model output

Methods

Public methods:

- `ModelExplainabilityCheckConfig$new()`
- `ModelExplainabilityCheckConfig$clone()`

Method `new()`**:** Initialize ModelExplainabilityCheckConfig class

Usage:

```
ModelExplainabilityCheckConfig$new(
  model_config,
  explainability_config,
  model_scores = NULL,
  ...
)
```

Arguments:

`model_config` (ModelConfig): Config of the model and its endpoint to be created.

`explainability_config` (SHAPConfig): Config of the specific explainability method. Currently, only SHAP is supported.

`model_scores` (str or int or ModelPredictedLabelConfig): Index or JSONPath location in the model output for the predicted scores to be explained (default: None). This is not required if the model output is a single score. Alternatively, an instance of ModelPredictedLabelConfig can be provided but this field CANNOT be any of PipelineNonPrimitiveInputTypes.

`...` : Parameters from ClarifyCheckConfig

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ModelExplainabilityCheckConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelQualityCheckConfig`

ModelQualityCheckConfig Class

Description

Model Quality Check Config.

Super class

sagemaker.workflow::QualityCheckConfig -> ModelQualityCheckConfig

Public fields

problem_type (str or PipelineNonPrimitiveInputTypes): The type of problem of this model quality monitoring. Valid values are "Regression", "BinaryClassification", "MulticlassClassification".

inference_attribute (str or PipelineNonPrimitiveInputTypes): Index or JSONpath to locate predicted label(s) (default: None).

probability_attribute (str or PipelineNonPrimitiveInputTypes): Index or JSONpath to locate probabilities (default: None).

ground_truth_attribute (str or PipelineNonPrimitiveInputTypes: Index or JSONpath to locate actual label(s) (default: None).

probability_threshold_attribute (str or PipelineNonPrimitiveInputTypes): Threshold to convert probabilities to binaries (default: None).

Methods

Public methods:

- ModelQualityCheckConfig$clone()

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ModelQualityCheckConfig$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

---

model_config *Export Airflow model config from a SageMaker model*

Description

Export Airflow model config from a SageMaker model

Usage

```
model_config(model, instance_type = NULL, role = NULL, image_uri = NULL)
```

Arguments

model	(sagemaker.model.Model): The Model object from which to export the Airflow config
instance_type	(str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'
role	(str): The "ExecutionRoleArn" IAM Role ARN for the model
image_uri	(str): An Docker image URI to use for deploying the model

Value

dict: Model config that can be directly used by SageMakerModelOperator in Airflow. It can also be part of the config used by SageMakerEndpointOperator and SageMakerTransformOperator in Airflow.

`model_config_from_estimator` *Export Airflow model config from a SageMaker estimator*

Description

Export Airflow model config from a SageMaker estimator

Usage

```
model_config_from_estimator(
  estimator,
  task_id,
  task_type,
  instance_type = NULL,
  role = NULL,
  image_uri = NULL,
  name = NULL,
  model_server_workers = NULL,
  vpc_config_override = "VPC_CONFIG_DEFAULT"
)
```

Arguments

`estimator`	(sagemaker.model.EstimatorBase): The SageMaker estimator to export Airflow config from. It has to be an estimator associated with a training job.
`task_id`	(str): The task id of any airflow.contrib.operators.SageMakerTrainingOperator or airflow.contrib.operators.SageMakerTuningOperator that generates training jobs in the DAG. The model config is built based on the training job generated in this operator.
`task_type`	(str): Whether the task is from SageMakerTrainingOperator or SageMakerTuningOperator. Values can be 'training', 'tuning' or None (which means training job is not from any task).
`instance_type`	(str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'
`role`	(str): The "ExecutionRoleArn" IAM Role ARN for the model
`image_uri`	(str): A Docker image URI to use for deploying the model
`name`	(str): Name of the model
`model_server_workers`	(int): The number of worker processes used by the inference server. If None, server will use one worker per vCPU. Only effective when estimator is a SageMaker framework.

vpc_config_override
: (dict[str, list[str]]): Override for VpcConfig set on the model. Default: use subnets and security groups from this Estimator. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

Value

dict: Model config that can be directly used by SageMakerModelOperator in Airflow. It can also be part of the config used by SageMakerEndpointOperator in Airflow.

ParallelismConfiguration

ParallelismConfiguration

Description

Parallelism config for SageMaker pipeline

Public fields

`max_parallel_execution_steps` Max number of steps which could be parallelized

Methods

Public methods:

- `ParallelismConfiguration$new()`
- `ParallelismConfiguration$to_request()`
- `ParallelismConfiguration$format()`
- `ParallelismConfiguration$clone()`

Method `new()`**:** Create a ParallelismConfiguration

Usage:

```
ParallelismConfiguration$new(max_parallel_execution_steps)
```

Arguments:

`max_parallel_execution_steps`, int: max number of steps which could be parallelized

Method `to_request()`**:** The request structure.

Usage:

```
ParallelismConfiguration$to_request()
```

Method `format()`**:** format class

Usage:

```
ParallelismConfiguration$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ParallelismConfiguration$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Parameter` *Workflow Parameter Class*

Description

Pipeline parameter for workflow.

Super class

`sagemaker.workflow::Entity` -> `Parameter`

Public fields

`name` The name of the parameter.

`parameter_type` The type of the parameter

`default_value` The default python value of the parameter

Active bindings

`expr` The 'Get' expression dict for a 'Parameter'

Methods

Public methods:

- `Parameter$new()`
- `Parameter$to_request()`
- `Parameter$clone()`

Method `new()`**:** Initialize Parameter class

Usage:

```
Parameter$new(
  name,
  parameter_type = ParameterTypeEnum$new(),
  default_value = NULL
)
```

Arguments:

`name` (str): The name of the parameter.

`parameter_type` (ParameterTypeEnum): The type of the parameter.

`default_value` (PrimitiveType): The default Python value of the parameter.

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
Parameter$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Parameter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ParameterBoolean` *Workflow ParameterBoolean class*

Description

Pipeline boolean parameter for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Parameter` -> `ParameterBoolean`

Methods

Public methods:

- `ParameterBoolean$new()`
- `ParameterBoolean$clone()`

Method `new()`**:** Create a pipeline boolean parameter.

Usage:

```
ParameterBoolean$new(name, default_value = NULL)
```

Arguments:

`name` (str): The name of the parameter.

`default_value` (str): The default Python value of the parameter. Defaults to None.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ParameterBoolean$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ParameterFloat *Workflow ParameterFloat class*

Description

Pipeline float parameter for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Parameter -> ParameterFloat

Active bindings

`float` Return default value or implicit value

Methods

Public methods:

- ParameterFloat$new()
- ParameterFloat$clone()

Method `new()`**:** Create a pipeline float parameter.

Usage:

```
ParameterFloat$new(name, default_value = NULL)
```

Arguments:

`name` (str): The name of the parameter.

`default_value` (float): The default Python value of the parameter.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ParameterFloat$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ParameterInteger *Workflow ParameterInteger class*

Description

Pipeline integer parameter for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Parameter -> ParameterInteger

Active bindings

int Return default value or implicit value

Methods

Public methods:

- ParameterInteger$new()
- ParameterInteger$clone()

Method new(): Create a pipeline integer parameter.

Usage:

```
ParameterInteger$new(name, default_value = NULL)
```

Arguments:

name (str): The name of the parameter.

default_value (int): The default Python value of the parameter.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ParameterInteger$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

ParameterString *Workflow ParameterString class*

Description

Pipeline string parameter for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Parameter` -> `ParameterString`

Public fields

`enum_values` Placeholder

Active bindings

`str` Return default value or implicit value

Methods

Public methods:

- `ParameterString$new()`
- `ParameterString$to_request()`
- `ParameterString$clone()`

Method `new()`**:** Create a pipeline string parameter.

Usage:

```
ParameterString$new(name, default_value = NULL, enum_values = NULL)
```

Arguments:

`name` (str): The name of the parameter.
`default_value` (str): The default Python value of the parameter.
`enum_values` (list): placeholder

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
ParameterString$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ParameterString$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Pipeline *Workflow Pipeline class*

Description

Pipeline for workflow.

Super class

`sagemaker.workflow::Entity` -> `Pipeline`

Methods

Public methods:

- `Pipeline$new()`
- `Pipeline$to_request()`
- `Pipeline$create()`
- `Pipeline$describe()`
- `Pipeline$update()`
- `Pipeline$upsert()`
- `Pipeline$delete()`
- `Pipeline$start()`
- `Pipeline$definition()`
- `Pipeline$clone()`

Method `new()`**:** Initialize Pipeline Class

Usage:

```
Pipeline$new(
  name,
  parameters = list(),

   pipeline_experiment_config = PipelineExperimentConfig$new(ExecutionVariables$PIPELINE_NAME,
    ExecutionVariables$PIPELINE_EXECUTION_ID),
  steps = list(),
  sagemaker_session = NULL
)
```

Arguments:

`name` (str): The name of the pipeline.

`parameters` (Sequence[Parameter]): The list of the parameters.

`pipeline_experiment_config` (Optional[PipelineExperimentConfig]): If set, the workflow will attempt to create an experiment and trial before executing the steps. Creation will be skipped if an experiment or a trial with the same name already exists. By default, pipeline name is used as experiment name and execution id is used as the trial name. If set to None, no experiment or trial will be created automatically.

`steps` (Sequence[Union[Step, StepCollection]]): The list of the non-conditional steps associated with the pipeline. Any steps that are within the 'if_steps' or 'else_steps' of a 'ConditionStep' cannot be listed in the steps of a pipeline. Of particular note, the workflow service rejects any pipeline definitions that specify a step in the list of steps of a pipeline and that step in the 'if_steps' or 'else_steps' of any 'ConditionStep'.

`sagemaker_session` (Session): Session object that manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the pipeline creates one using the default AWS configuration chain.

Method `to_request()`**:** Gets the request structure for workflow service calls.

Usage:

```
Pipeline$to_request()
```

Method `create()`**:** Creates a Pipeline in the Pipelines service.

Usage:

```
Pipeline$create(
  role_arn,
  description = NULL,
  tags = NULL,
  parallelism_config = NULL
)
```

Arguments:

`role_arn` (str): The role arn that is assumed by the pipeline to create step artifacts.

`description` (str): A description of the pipeline.

`tags` (List[Dict[str, str]]): A list of "Key": "string", "Value": "string" dicts as tags.

`parallelism_config` (Optional[ParallelismConfiguration]): Parallelism configuration that is applied to each of the executions of the pipeline. It takes precedence over the parallelism configuration of the parent pipeline.

Returns: A response dict from the service.

Method `describe()`**:** Describes a Pipeline in the Workflow service.

Usage:

```
Pipeline$describe()
```

Returns: Response dict from the service. See 'boto3 client documentation https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sagemaker.html#SageMaker.Client.describe_pipeline

Method `update()`**:** Updates a Pipeline in the Workflow service.

Usage:

```
Pipeline$update(role_arn, description = NULL, parallelism_config = NULL)
```

Arguments:

`role_arn` (str): The role arn that is assumed by pipelines to create step artifacts.

`description` (str): A description of the pipeline.

`parallelism_config` (Optional[ParallelismConfiguration]): Parallelism configuration that is applied to each of the executions of the pipeline. It takes precedence over the parallelism configuration of the parent pipeline.

Returns: A response dict from the service.

Method `upsert()`**:** Creates a pipeline or updates it, if it already exists.

Usage:

```
Pipeline$upsert(
  role_arn,
  description = NULL,
  tags = NULL,
  parallelism_config = NULL
)
```

Arguments:

`role_arn` (str): The role arn that is assumed by workflow to create step artifacts.

`description` (str): A description of the pipeline.

`tags` (List[Dict[str, str]]): A list of "Key": "string", "Value": "string" dicts as tags.

`parallelism_config` (Optional[Config for parallel steps, Parallelism configuration that is applied to each of. the executions

Returns: response dict from service

Method `delete()`**:** Deletes a Pipeline in the Workflow service.

Usage:

```
Pipeline$delete()
```

Returns: A response dict from the service.

Method `start()`**:** Starts a Pipeline execution in the Workflow service.

Usage:

```
Pipeline$start(
  parameters = NULL,
  execution_display_name = NULL,
  execution_description = NULL,
  parallelism_config = NULL
)
```

Arguments:

`parameters` (Dict[str, Union[str, bool, int, float]]): values to override pipeline parameters.

`execution_display_name` (str): The display name of the pipeline execution.

`execution_description` (str): A description of the execution.

`parallelism_config` (Optional[ParallelismConfiguration]): Parallelism configuration that is applied to each of the executions of the pipeline. It takes precedence over the parallelism configuration of the parent pipeline.

Returns: A ‘.PipelineExecution‘ instance, if successful.

Method `definition()`**:** Converts a request structure to string representation for workflow service calls.

Usage:

```
Pipeline$definition()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Pipeline$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PipelineExperimentConfig` *Workflow PipelineExperimentConfig class*

Description

Experiment config for SageMaker pipeline.

Super class

`sagemaker.workflow::Entity` -> `PipelineExperimentConfig`

Methods

Public methods:

- `PipelineExperimentConfig$new()`
- `PipelineExperimentConfig$to_request()`
- `PipelineExperimentConfig$clone()`

Method `new()`**:** Create a PipelineExperimentConfig

Usage:

```
PipelineExperimentConfig$new(experiment_name, trial_name)
```

Arguments:

`experiment_name` (Union[str, Parameter, ExecutionVariable, Expression]): the name of the experiment that will be created.

`trial_name` (Union[str, Parameter, ExecutionVariable, Expression]): the name of the trial that will be created.

Examples:

```
# Use pipeline name as the experiment name and pipeline execution id as the trial name::
PipelineExperimentConfig$new(
   ExecutionVariables$PIPELINE_NAME, ExecutionVariables$PIPELINE_EXECUTION_ID)
# Use a customized experiment name and pipeline execution id as the trial name::
PipelineExperimentConfig$new(
     'MyExperiment', ExecutionVariables$PIPELINE_EXECUTION_ID)
```

Method `to_request()`**:** Returns: the request structure.

Usage:

```
PipelineExperimentConfig$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PipelineExperimentConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `PipelineExperimentConfig$new`
## ------------------------------------------------

# Use pipeline name as the experiment name and pipeline execution id as the trial name::
PipelineExperimentConfig$new(
     ExecutionVariables$PIPELINE_NAME, ExecutionVariables$PIPELINE_EXECUTION_ID)
# Use a customized experiment name and pipeline execution id as the trial name::
PipelineExperimentConfig$new(
     'MyExperiment', ExecutionVariables$PIPELINE_EXECUTION_ID)
```

PipelineExperimentConfigProperties

Workflow PipelineExperimentConfigProperties enum like class

Description

Enum-like class for all pipeline experiment config property references.

Usage

```
PipelineExperimentConfigProperties
```

Format

An object of class `Enum` (inherits from `environment`) of length 2.

PipelineExperimentConfigProperty
Workflow PipelineExperimentConfigProperty class

Description

Reference to pipeline experiment config property.

Super class

`sagemaker.workflow::Expression` -> `PipelineExperimentConfigProperty`

Active bindings

`expr` The 'Get' expression dict for a pipeline experiment config property.

Methods

Public methods:

- `PipelineExperimentConfigProperty$new()`
- `PipelineExperimentConfigProperty$clone()`

Method `new()`**:** Create a reference to pipeline experiment property.

Usage:

```
PipelineExperimentConfigProperty$new(name)
```

Arguments:

`name` (str): The name of the pipeline experiment config property.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PipelineExperimentConfigProperty$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PipelineVariable` *Base object for pipeline variables*

Description

PipelineVariables must implement the expr property.

Super class

`sagemaker.workflow::PropertiesMeta` -> `PipelineVariable`

Active bindings

`expr` Get the expression structure for workflow service calls.

Methods

Public methods:

- `PipelineVariable$to_string()`
- `PipelineVariable$startswith()`
- `PipelineVariable$endswith()`
- `PipelineVariable$clone()`

Method `to_string()`**:** Prompt the pipeline to convert the pipeline variable to String in runtime

Usage:

```
PipelineVariable$to_string()
```

Method `startswith()`**:** Simulate the Python string's built-in method: startswith

Usage:

```
PipelineVariable$startswith(prefix, start = NULL, end = NULL)
```

Arguments:

`prefix` (str, tuple): The (tuple of) string to be checked.
`start` (int): To set the start index of the matching boundary (default: None).
`end` (int): To set the end index of the matching boundary (default: None).

Returns: bool: Always return False as Pipeline variables are parsed during execution runtime

Method `endswith()`**:** Simulate the Python string's built-in method: endswith

Usage:

```
PipelineVariable$endswith(suffix, start = NULL, end = NULL)
```

Arguments:

`suffix` (str, tuple): The (tuple of) string to be checked.
`start` (int): To set the start index of the matching boundary (default: None).
`end` (int): To set the end index of the matching boundary (default: None).

Returns: bool: Always return False as Pipeline variables are parsed during execution runtime

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PipelineVariable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`prepare_amazon_algorithm_estimator` *Sets up amazon algorithm estimator.*

Description

This is done by adding the required 'feature_dim' hyperparameter from training data.

Usage

```
prepare_amazon_algorithm_estimator(estimator, inputs, mini_batch_size = NULL)
```

Arguments

`estimator`	(sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase): An estimator for a built-in Amazon algorithm to get information from and update.
`inputs`	: The training data. * (sagemaker.amazon.amazon_estimator.RecordSet) - A collection of Amazon :class:~'Record' objects serialized and stored in S3. For use with an estimator for an Amazon algorithm. * (list[sagemaker.amazon.amazon_estimator.RecordSet]) - A list of :class:~'sagemaker.amazon.amazon_estimator.RecordSet' objects, where each instance is a different channel of training data.
`mini_batch_size`	(numeric):

`prepare_framework` *Prepare S3 operations and environment variables related to framework.*

Description

S3 operations specify where to upload 'source_dir'.

Usage

```
prepare_framework(estimator, s3_operations)
```

Arguments

estimator	(sagemaker.estimator.Estimator): The framework estimator to get information from and update.
s3_operations	(list): The dict to specify s3 operations (upload 'source_dir').

---

prepare_framework_container_def

This prepares the framework model container information and specifies related S3 operations.

Description

Prepare the framework model container information. Specify related S3 operations for Airflow to perform. (Upload 'source_dir')

Usage

```
prepare_framework_container_def(model, instance_type, s3_operations)
```

Arguments

model	(sagemaker.model.FrameworkModel): The framework model
instance_type	(str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.
s3_operations	(dict): The dict to specify S3 operations (upload 'source_dir').

Value

dict: The container information of this framework model.

ProcessingStep *Workflow ProcessingStep Class*

Description

Processing step for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Step -> sagemaker.workflow::ConfigurableRetryStep -> ProcessingStep

Active bindings

`arguments` The arguments dict that is used to call 'create_processing_job'. NOTE: The CreateProcessingJob request is not quite the args list that workflow needs. ProcessingJobName and ExperimentConfig cannot be included in the arguments.

`properties` A Properties object representing the DescribeProcessingJobResponse data model.

Methods

Public methods:

- `ProcessingStep$new()`
- `ProcessingStep$to_request()`
- `ProcessingStep$clone()`

Method `new()`**:** Construct a ProcessingStep, given a 'Processor' instance. In addition to the processor instance, the other arguments are those that are supplied to the 'process' method of the 'sagemaker.processing.Processor'.

Usage:

```
ProcessingStep$new(
  name,
  processor,
  display_name = NULL,
  description = NULL,
  inputs = NULL,
  outputs = NULL,
  job_arguments = NULL,
  code = NULL,
  property_files = NULL,
  cache_config = NULL,
  depends_on = NULL,
  retry_policies = NULL,
  kms_key = NULL
)
```

Arguments:

`name` (str): The name of the processing step.

`processor` (Processor): A 'sagemaker.processing.Processor' instance.

`display_name` (str): The display name of the processing step.

`description` (str): The description of the processing step.

`inputs` (List[ProcessingInput]): A list of 'sagemaker.processing.ProcessorInput' instances. Defaults to 'None'.

`outputs` (List[ProcessingOutput]): A list of 'sagemaker.processing.ProcessorOutput' instances. Defaults to 'None'.

`job_arguments` (List[str]): A list of strings to be passed into the processing job. Defaults to 'None'.

`code` (str): This can be an S3 URI or a local path to a file with the framework script to run. Defaults to 'None'.

property_files (List[PropertyFile]): A list of property files that workflow looks for and resolves from the configured processing output list.

cache_config (CacheConfig): A ‘sagemaker.workflow.steps.CacheConfig‘ instance.

depends_on (List[str] or List[Step]): A list of step names or step instance this ‘sagemaker.workflow.steps.ProcessingStep depends on

retry_policies (List[RetryPolicy]): A list of retry policy

kms_key (str): The ARN of the KMS key that is used to encrypt the user code file. Defaults to ‘None‘.

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
ProcessingStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ProcessingStep$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`processing_config` *Export Airflow processing config from a SageMaker processor*

Description

Export Airflow processing config from a SageMaker processor

Usage

```
processing_config(
  processor,
  inputs = NULL,
  outputs = NULL,
  job_name = NULL,
  experiment_config = NULL,
  container_arguments = NULL,
  container_entrypoint = NULL,
  kms_key_id = NULL
)
```

Arguments

processor	(sagemaker.processor.Processor): The SageMaker processor to export Airflow config from.
inputs	(list[:class:‘~sagemaker.processing.ProcessingInput‘]): Input files for the processing job. These must be provided as :class:‘~sagemaker.processing.ProcessingInput‘ objects (default: None).

outputs (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): Outputs for the processing job. These can be specified as either path strings or :class:‘~sagemaker.processing.ProcessingOutp objects (default: None).

job_name (str): Processing job name. If not specified, the processor generates a default job name, based on the base job name and current timestamp.

experiment_config (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: ’ExperimentName’, ’TrialName’, and ’TrialComponentDisplayName’.

container_arguments ([str]): The arguments for a container used to run a processing job.

container_entrypoint ([str]): The entrypoint for a container used to run a processing job.

kms_key_id (str): The AWS Key Management Service (AWS KMS) key that Amazon SageMaker uses to encrypt the processing job output. KmsKeyId can be an ID of a KMS key, ARN of a KMS key, alias of a KMS key, or alias of a KMS key. The KmsKeyId is applied to all outputs.

Value

dict: Processing config that can be directly used by SageMakerProcessingOperator in Airflow.

Properties *Workflow Properties Class*

Description

Properties for use in workflow expressions.

Super classes

sagemaker.workflow::PropertiesMeta -> sagemaker.workflow::PipelineVariable -> Properties

Public fields

shape_name The botocore sagemaker service model shape name.

shape_names A List of the botocore sagemaker service model shape name

Active bindings

expr The ’Get’ expression dict for a ‘Properties‘.

Methods

Public methods:

- Properties$new()
- Properties$clone()

Method `new()`**:** Create a Properties instance representing the given shape.

Usage:

```
Properties$new(
  path,
  shape_name = NULL,
  shape_names = NULL,
  service_name = "sagemaker"
)
```

Arguments:

`path` (str): The parent path of the Properties instance.

`shape_name` (str): The botocore sagemaker service model shape name.

`shape_names` (str): A List of the botocore sagemaker service model shape name.

`service_name` (str):

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Properties$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

PropertiesList *Workflow PropertiesList class*

Description

PropertiesList for use in workflow expressions

Super classes

sagemaker.workflow::PropertiesMeta -> sagemaker.workflow::PipelineVariable -> sagemaker.workflow::Prop
-> PropertiesList

Methods

Public methods:

- PropertiesList$new()
- PropertiesList$get_item()
- PropertiesList$clone()

Method `new()`**:** Create a PropertiesList instance representing the given shape.

Usage:

```
PropertiesList$new(path, shape_name = NULL, service_name = "sagemaker")
```

Arguments:

`path` (str): The parent path of the PropertiesList instance.

`shape_name` (str): The botocore sagemaker service model shape name.

`service_name` (str): The botocore service name.

`root_shape_name` (str): The botocore sagemaker service model shape name.

Method `get_item()`**:** Populate the indexing item with a Property, for both lists and dictionaries.

Usage:

```
PropertiesList$get_item(item)
```

Arguments:

`item` (Union[int, str]): The index of the item in sequence.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PropertiesList$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PropertiesMap` *PropertiesMap class*

Description

PropertiesMap for use in workflow expressions.

Super classes

`sagemaker.workflow::PropertiesMeta` -> `sagemaker.workflow::PipelineVariable` -> `sagemaker.workflow::Prop`
-> `PropertiesMap`

Public fields

`path` The parent path of the PropertiesMap instance.

`shape_name` The botocore sagemaker service model shape name.

`service_name` The botocore service name.

Methods

Public methods:

- PropertiesMap$new()
- PropertiesMap$get_item()
- PropertiesMap$clone()

Method `new()`**:** Create a PropertiesMap instance representing the given shape.

Usage:

```
PropertiesMap$new(path, shape_name = NULL, service_name = "sagemaker")
```

Arguments:

`path` (str): The parent path of the PropertiesMap instance.

`shape_name` (str): The botocore sagemaker service model shape name.

`service_name` (str): The botocore service name.

Method `get_item()`**:** Populate the indexing item with a Property, for both lists and dictionaries.

Usage:

```
PropertiesMap$get_item(item)
```

Arguments:

`item` (Union[int, str]): The index of the item in sequence.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PropertiesMap$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PropertyFile` *PropertyFile Class*

Description

Provides a property file struct.

Super class

`sagemaker.workflow::Expression` -> `PropertyFile`

Public fields

`name` The name of the property file for reference with 'JsonGet' functions.

`output_name` The name of the processing job output channel.

`path` The path to the file at the output channel location.

Active bindings

`expr` Get the expression structure for workflow service calls.

Methods

Public methods:

- `PropertyFile$new()`
- `PropertyFile$clone()`

Method `new()`**:** Initializing PropertyFile Class

Usage:

```
PropertyFile$new(name, output_name, path)
```

Arguments:

`name` (str): The name of the property file for reference with 'JsonGet' functions.

`output_name` (str): The name of the processing job output channel.

`path` (str): The path to the file at the output channel location.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PropertyFile$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`QualityCheckConfig` *QualityCheckConfig class*

Description

Quality Check Config.

Public fields

`baseline_dataset` str or PipelineNonPrimitiveInputTypes): The path to the baseline_dataset file. This can be a local path or an S3 uri string

`dataset_format` (dict): The format of the baseline_dataset.

`output_s3_uri` (str or PipelineNonPrimitiveInputTypes): Desired S3 destination of the constraint_violations and statistics json files (default: None). If not specified an auto generated path will be used: "s3://<default_session_bucket>/model-monitor/baselining/<job_name>/results"

`post_analytics_processor_script` (str): The path to the record post-analytics processor script (default: None). This can be a local path or an S3 uri string but CANNOT be any of PipelineNonPrimitiveInputTypes.

Methods

Public methods:

- QualityCheckConfig$clone()

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
QualityCheckConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`QualityCheckStep` *QualityCheckStep class*

Description

QualityCheck step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `QualityCheckStep`

Active bindings

`arguments` The arguments dict that is used to define the QualityCheck step.

`properties` A Properties object representing the output parameters of the QualityCheck step.

Methods

Public methods:

- QualityCheckStep$new()
- QualityCheckStep$to_request()
- QualityCheckStep$clone()

Method `new()`**:** Constructs a QualityCheckStep.

Usage:

```
QualityCheckStep$new(
  name,
  quality_check_config,
  check_job_config,
  skip_check = FALSE,
  register_new_baseline = FALSE,
  model_package_group_name = NULL,
  supplied_baseline_statistics = NULL,
  supplied_baseline_constraints = NULL,
```

```
  display_name = NULL,
  description = NULL,
  cache_config = NULL,
  depends_on = NULL
)
```

Arguments:

`name` (str): The name of the QualityCheckStep step.

`quality_check_config` (QualityCheckConfig): A QualityCheckConfig instance.

`check_job_config` (CheckJobConfig): A CheckJobConfig instance.

`skip_check` (bool or PipelineNonPrimitiveInputTypes): Whether the check should be skipped (default: False).

`register_new_baseline` (bool or PipelineNonPrimitiveInputTypes): Whether the new baseline should be registered (default: False).

`model_package_group_name` (str or PipelineNonPrimitiveInputTypes): The name of a registered model package group, among which the baseline will be fetched from the latest approved model (default: None).

`supplied_baseline_statistics` (str or PipelineNonPrimitiveInputTypes): The S3 path to the supplied statistics object representing the statistics JSON file which will be used for drift to check (default: None).

`supplied_baseline_constraints` (str or PipelineNonPrimitiveInputTypes): The S3 path to the supplied constraints object representing the constraints JSON file which will be used for drift to check (default: None).

`display_name` (str): The display name of the QualityCheckStep step (default: None).

`description` (str): The description of the QualityCheckStep step (default: None).

`cache_config` (CacheConfig): A ‘sagemaker.workflow.steps.CacheConfig‘ instance (default: None).

`depends_on` (List[str] or List[Step]): A list of step names or step instances this ‘sagemaker.workflow.steps.QualityCheck depends on (default: None).

Method `to_request()`**:** Updates the dictionary with cache configuration etc.

Usage:

```
QualityCheckStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
QualityCheckStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

RegisterModel *Workflow RegisterModel class*

Description

Register Model step collection for workflow.

Super class

sagemaker.workflow::StepCollection -> RegisterModel

Methods

Public methods:

- RegisterModel$new()
- RegisterModel$clone()

Method new(): Construct steps '_RepackModelStep' and '_RegisterModelStep' based on the estimator.

Usage:

```
RegisterModel$new(
  name,
  content_types,
  response_types,
  inference_instances,
  transform_instances,
  estimator = NULL,
  model_data = NULL,
  depends_on = NULL,
  repack_model_step_retry_policies = NULL,
  register_model_step_retry_policies = NULL,
  model_package_group_name = NULL,
  model_metrics = NULL,
  approval_status = NULL,
  image_uri = NULL,
  compile_model_family = NULL,
  display_name = NULL,
  description = NULL,
  tags = NULL,
  model = NULL,
  drift_check_baselines = NULL,
  ...
)
```

Arguments:

name (str): The name of the training step.

`content_types` (list): The supported MIME types for the input data (default: None).

`response_types` (list): The supported MIME types for the output data (default: None).

`inference_instances` (list): A list of the instance types that are used to generate inferences in real-time (default: None).

`transform_instances` (list): A list of the instance types on which a transformation job can be run or on which an endpoint can be deployed (default: None).

`estimator` The estimator instance.

`model_data` The S3 uri to the model data from training.

`depends_on` (List[str] or List[Step]): The list of step names or step instances the first step in the collection depends on

`repack_model_step_retry_policies` (List[RetryPolicy]): The list of retry policies for the repack model step

`register_model_step_retry_policies` (List[RetryPolicy]): The list of retry policies for register model step

`model_package_group_name` (str): The Model Package Group name, exclusive to ‘model_package_name‘, using ‘model_package_group_name‘ makes the Model Package versioned (default: None).

`model_metrics` (ModelMetrics): ModelMetrics object (default: None).

`approval_status` (str): Model Approval Status, values can be "Approved", "Rejected", or "PendingManualApproval" (default: "PendingManualApproval").

`image_uri` (str): The container image uri for Model Package, if not specified, Estimator's training container image is used (default: None).

`compile_model_family` (str): The instance family for the compiled model. If specified, a compiled model is used (default: None).

`display_name` (str): The display name of the step.

`description` (str): Model Package description (default: None).

`tags` (List[dict[str, str]]): The list of tags to attach to the model package group. Note that tags will only be applied to newly created model package groups; if the name of an existing group is passed to "model_package_group_name", tags will not be applied.

`model` (object or Model): A PipelineModel object that comprises a list of models which gets executed as a serial inference pipeline or a Model object.

`drift_check_baselines` (DriftCheckBaselines): DriftCheckBaselines object (default: None).

`...` : additional arguments to ‘create_model‘.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RegisterModel$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

RetryPolicy *RetryPolicy base class*

Description

RetryPolicy base class

RetryPolicy base class

Super class

`sagemaker.workflow::Entity` -> `RetryPolicy`

Public fields

`backoff_rate` (float): The multiplier by which the retry interval increases during each attempt (default: 2.0)

`interval_seconds` (int): An integer that represents the number of seconds before the first retry attempt (default: 1)

`max_attempts` (int): A positive integer that represents the maximum number of retry attempts. (default: None)

`expire_after_mins` (int): A positive integer that represents the maximum minute to expire any further retry attempt (default: None)

Methods

Public methods:

- `RetryPolicy$new()`
- `RetryPolicy$validate_backoff_rate()`
- `RetryPolicy$validate_interval_seconds()`
- `RetryPolicy$validate_max_attempts()`
- `RetryPolicy$validate_expire_after_mins()`
- `RetryPolicy$to_request()`
- `RetryPolicy$format()`
- `RetryPolicy$clone()`

Method `new()`**:** Initialize RetryPolicy class

Usage:

```
RetryPolicy$new(
  backoff_rate = DEFAULT_BACKOFF_RATE,
  interval_seconds = DEFAULT_INTERVAL_SECONDS,
  max_attempts = NULL,
  expire_after_mins = NULL
)
```

Arguments:

`backoff_rate` (float): The multiplier by which the retry interval increases during each attempt (default: 2.0)

`interval_seconds` (int): An integer that represents the number of seconds before the first retry attempt (default: 1)

`max_attempts` (int): A positive integer that represents the maximum number of retry attempts. (default: None)

`expire_after_mins` (int): A positive integer that represents the maximum minute to expire any further retry attempt (default: None)

Method `validate_backoff_rate()`**:** Validate the input back off rate type

Usage:

```
RetryPolicy$validate_backoff_rate(value)
```

Arguments:

`value` object to be checked

Method `validate_interval_seconds()`**:** Validate the input interval seconds

Usage:

```
RetryPolicy$validate_interval_seconds(value)
```

Arguments:

`value` object to be checked

Method `validate_max_attempts()`**:** Validate the input max attempts

Usage:

```
RetryPolicy$validate_max_attempts(value)
```

Arguments:

`value` object to be checked

Method `validate_expire_after_mins()`**:** Validate expire after mins

Usage:

```
RetryPolicy$validate_expire_after_mins(value)
```

Arguments:

`value` object to be checked

Method `to_request()`**:** Get the request structure for workflow service calls.

Usage:

```
RetryPolicy$to_request()
```

Arguments:

`value` object to be checked

Method `format()`**:** format class

Usage:

```
RetryPolicy$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RetryPolicy$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`SageMakerJobExceptionTypeEnum`
SageMaker Job ExceptionType enum.

Description

SageMaker Job ExceptionType enum.

Usage

```
SageMakerJobExceptionTypeEnum
```

Format

An object of class `Enum` (inherits from `environment`) of length 3.

`SageMakerJobStepRetryPolicy`
SageMakerJobStepRetryPolicy class

Description

RetryPolicy for exception thrown by SageMaker Job.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::RetryPolicy` -> `SageMakerJobStepRetryPolicy`

Public fields

`exception_type_list` Contains exception_types or failure_reason_types

Methods

Public methods:

- SageMakerJobStepRetryPolicy$new()
- SageMakerJobStepRetryPolicy$to_request()
- SageMakerJobStepRetryPolicy$clone()

Method `new()`**:** Initialize SageMakerJobStepRetryPolicy

Usage:

```
SageMakerJobStepRetryPolicy$new(
  exception_types = NULL,
  failure_reason_types = NULL,
  backoff_rate = 2,
  interval_seconds = 1,
  max_attempts = NULL,
  expire_after_mins = NULL
)
```

Arguments:

`exception_types` (List[SageMakerJobExceptionTypeEnum]): The SageMaker exception to match for this policy. The SageMaker exceptions captured here are the exceptions thrown by synchronously creating the job. For instance the resource limit exception.

`failure_reason_types` (List[SageMakerJobExceptionTypeEnum]): the SageMaker failure reason types to match for this policy. The failure reason type is presented in FailureReason field of the Describe response, it indicates the runtime failure reason for a job.

`backoff_rate` (float): The multiplier by which the retry interval increases during each attempt (default: 2.0)

`interval_seconds` (int): An integer that represents the number of seconds before the first retry attempt (default: 1)

`max_attempts` (int): A positive integer that represents the maximum number of retry attempts. (default: None)

`expire_after_mins` (int): A positive integer that represents the maximum minute to expire any further retry attempt (default: None)

Method `to_request()`**:** Gets the request structure for retry policy

Usage:

```
SageMakerJobStepRetryPolicy$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SageMakerJobStepRetryPolicy$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Step *Workflow Step class*

Description

Pipeline step for workflow.

Super class

`sagemaker.workflow::Entity` -> `Step`

Public fields

`name` The name of the step.

`display_name` The display name of the step.

`description` The description of the step.

`step_type` The type of the step.

`depends_on` The list of step names the current step depends on

`retry_policies` (List[RetryPolicy]): The custom retry policy configuration

Active bindings

`arguments` The arguments to the particular step service call.

`properties` The properties of the particular step.

`ref` Gets a reference dict for steps

Methods

Public methods:

- `Step$new()`
- `Step$to_request()`
- `Step$add_depends_on()`
- `Step$format()`
- `Step$clone()`

Method `new()`**:** Initialize Workflow Step

Usage:

```
Step$new(
  name,
  display_name = NULL,
  description = NULL,
  step_type = enum_items(StepTypeEnum),
  depends_on = NULL
)
```

Arguments:

`name` (str): The name of the step.

`display_name` (str): The display name of the step.

`description` (str): The description of the step.

`step_type` (StepTypeEnum): The type of the step.

`depends_on` (List[str] or List[Step]): The list of step names or step instances the current step depends on

Method `to_request()`**:** Gets the request structure for workflow service calls.

Usage:

```
Step$to_request()
```

Method `add_depends_on()`**:** Add step names to the current step depends on list

Usage:

```
Step$add_depends_on(step_names)
```

Arguments:

`step_names` (list): placeholder

Method `format()`**:** formats class

Usage:

```
Step$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Step$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`StepCollection` *Workflow StepCollection class*

Description

A wrapper of pipeline steps for workflow.

Public fields

`steps` A list of steps.

Methods

Public methods:

- `StepCollection$new()`
- `StepCollection$request_list()`
- `StepCollection$format()`
- `StepCollection$clone()`

Method `new()`**:** Initialize StepCollection class

Usage:

```
StepCollection$new(steps)
```

Arguments:

`steps` (List[Step]): A list of steps.

Method `request_list()`**:** Get the request structure for workflow service calls.

Usage:

```
StepCollection$request_list()
```

Method `format()`**:** format class

Usage:

```
StepCollection$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
StepCollection$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`StepExceptionTypeEnum` *Step ExceptionType enum.*

Description

Step ExceptionType enum.

Usage

```
StepExceptionTypeEnum
```

Format

An object of class `Enum` (inherits from `environment`) of length 2.

StepRetryPolicy *StepRetryPolicy class*

Description

RetryPolicy for a retryable step. The pipeline service will retry

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::RetryPolicy` -> `StepRetryPolicy`

Public fields

`exception_types` (List[StepExceptionTypeEnum]): the exception types to match for this policy

Methods

Public methods:

- `StepRetryPolicy$new()`
- `StepRetryPolicy$to_request()`
- `StepRetryPolicy$clone()`

Method `new()`**:** Initialize StepRetryPolicy class

Usage:

```
StepRetryPolicy$new(
  exception_types,
  backoff_rate = 2,
  interval_seconds = 1,
  max_attempts = NULL,
  expire_after_mins = NULL
)
```

Arguments:

`exception_types` (List[StepExceptionTypeEnum]): the exception types to match for this policy

`backoff_rate` (float): The multiplier by which the retry interval increases during each attempt (default: 2.0)

`interval_seconds` (int): An integer that represents the number of seconds before the first retry attempt (default: 1)

`max_attempts` (int): A positive integer that represents the maximum number of retry attempts. (default: None)

`expire_after_mins` (int): A positive integer that represents the maximum minute to expire any further retry attempt (default: None)

Method `to_request()`**:** Gets the request structure for retry policy.

Usage:

```
StepRetryPolicy$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
StepRetryPolicy$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`StepTypeEnum` *Workflow StepTypeEnum class*

Description

Enum of step types.

Usage

```
StepTypeEnum
```

Format

An object of class `Enum` (inherits from `environment`) of length 13.

`TrainingStep` *Workflow TraingingStep class*

Description

Training step for workflow.

Super classes

`sagemaker.workflow::Entity` -> `sagemaker.workflow::Step` -> `sagemaker.workflow::ConfigurableRetryStep` -> `TrainingStep`

Active bindings

`arguments` The arguments dict that is used to call 'create_training_job'. NOTE: The CreateTrainingJob request is not quite the args list that workflow needs. The TrainingJobName and ExperimentConfig attributes cannot be included.

`properties` A Properties object representing the DescribeTrainingJobResponse data model.

Methods

Public methods:

- TrainingStep$new()
- TrainingStep$to_request()
- TrainingStep$clone()

Method `new()`**:** Construct a TrainingStep, given an 'EstimatorBase' instance. In addition to the estimator instance, the other arguments are those that are supplied to the 'fit' method of the 'sagemaker.estimator.Estimator'.

Usage:

```
TrainingStep$new(
  name,
  estimator,
  display_name = NULL,
  description = NULL,
  inputs = NULL,
  cache_config = NULL,
  depends_on = NULL,
  retry_policies = NULL
)
```

Arguments:

`name` (str): The name of the training step.

`estimator` (EstimatorBase): A 'sagemaker.estimator.EstimatorBase' instance.

`display_name` (str): The display name of the training step.

`description` (str): The description of the training step.

`inputs` (str or dict or sagemaker.inputs.TrainingInput or sagemaker.inputs.FileSystemInput): Information about the training data. This can be one of three types:

- (str) the S3 location where training data is saved, or a file:// path in local mode.
- (dict[str, str] or dict[str, sagemaker.inputs.TrainingInput]) If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:'~sagemaker.inputs.TrainingInput' objects.
- (sagemaker.inputs.TrainingInput) - channel configuration for S3 data sources that can provide additional information as well as the path to the training dataset. See :func:'sagemaker.inputs.TrainingInput for full details.
- (sagemaker.inputs.FileSystemInput) - channel configuration for a file system data source that can provide additional information as well as the path to the training dataset.

`cache_config` (CacheConfig): A 'sagemaker.workflow.steps.CacheConfig' instance.

`depends_on` (List[str]): A list of step names this 'sagemaker.workflow.steps.TrainingStep' depends on

`retry_policies` (List[RetryPolicy]): A list of retry policy

Method `to_request()`**:** A Properties object representing the DescribeTrainingJobResponse data model.

Usage:

```
TrainingStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TrainingStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`training_base_config` *Export Airflow base training config from an estimator*

Description

Export Airflow base training config from an estimator

Usage

```
training_base_config(
  estimator,
  inputs = NULL,
  job_name = NULL,
  mini_batch_size = NULL
)
```

Arguments

`estimator`	(sagemaker.estimator.EstimatorBase): The estimator to export training config from. Can be a BYO estimator, Framework estimator or Amazon algorithm estimator.
`inputs`	: Information about the training data. Please refer to the "fit()" method of the associated estimator, as this can take any of the following forms: * (str) - The S3 location where training data is saved. * (dict[str, str] or dict[str, sagemaker.inputs.TrainingInput]) - If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:`~sagemaker.inputs.TrainingInput` objects. * (sagemaker.inputs.TrainingInput) - Channel configuration for S3 data sources that can provide additional information about the training dataset. See :func:`sagemaker.inputs.TrainingInput` for full details. * (sagemaker.amazon.amazon_estimator.RecordS - A collection of Amazon :class:~`Record` objects serialized and stored in S3. For use with an estimator for an Amazon algorithm. * (list[sagemaker.amazon.amazon_estimator.RecordS - A list of :class:~`sagemaker.amazon.amazon_estimator.RecordSet` objects, where each instance is a different channel of training data.
`job_name`	(str): Specify a training job name if needed.
`mini_batch_size`	(int): Specify this argument only when estimator is a built-in estimator of an Amazon algorithm. For other estimators, batch size should be specified in the estimator.

Value

dict: Training config that can be directly used by SageMakerTrainingOperator in Airflow.

`training_config` *Export Airflow training config from an estimator*

Description

Export Airflow training config from an estimator

Usage

```
training_config(
  estimator,
  inputs = NULL,
  job_name = NULL,
  mini_batch_size = NULL
)
```

Arguments

`estimator`	(sagemaker.estimator.EstimatorBase): The estimator to export training config from. Can be a BYO estimator, Framework estimator or Amazon algorithm estimator.
`inputs`	: Information about the training data. Please refer to the "fit()" method of the associated estimator, as this can take any of the following forms: * (str) - The S3 location where training data is saved. * (dict[str, str] or dict[str, sagemaker.inputs.TrainingInput]) - If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:`~sagemaker.inputs.TrainingInput` objects. * (sagemaker.inputs.TrainingInput) - Channel configuration for S3 data sources that can provide additional information about the training dataset. See :func:`sagemaker.inputs.TrainingInput` for full details. * (sagemaker.amazon.amazon_estimator.RecordS - A collection of Amazon :class:~`Record` objects serialized and stored in S3. For use with an estimator for an Amazon algorithm. * (list[sagemaker.amazon.amazon_estimator.RecordS - A list of :class:~`sagemaker.amazon.amazon_estimator.RecordSet` objects, where each instance is a different channel of training data.
`job_name`	(str): Specify a training job name if needed.
`mini_batch_size`	(int): Specify this argument only when estimator is a built-in estimator of an Amazon algorithm. For other estimators, batch size should be specified in the estimator.

Value

list: Training config that can be directly used by SageMakerTrainingOperator in Airflow.

TransformStep *Workflow TransformStep class*

Description

Transform step for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Step -> sagemaker.workflow::ConfigurableRetryStep -> TransformStep

Active bindings

`arguments` The arguments dict that is used to call 'create_transform_job'. NOTE: The CreateTransformJob request is not quite the args list that workflow needs. TransformJobName and ExperimentConfig cannot be included in the arguments.

`properties` A Properties object representing the DescribeTransformJobResponse data model.

Methods

Public methods:

- `TransformStep$new()`
- `TransformStep$to_request()`
- `TransformStep$clone()`

Method `new()`**:** Constructs a TransformStep, given an 'Transformer' instance. In addition to the transformer instance, the other arguments are those that are supplied to the 'transform' method of the 'sagemaker.transformer.Transformer'.

Usage:

```
TransformStep$new(
  name,
  transformer,
  inputs,
  display_name = NULL,
  description = NULL,
  cache_config = NULL,
  depends_on = NULL,
  retry_policies = NULL
)
```

Arguments:

`name` (str): The name of the transform step.

`transformer` (Transformer): A 'sagemaker.transformer.Transformer' instance.

`inputs` (TransformInput): A 'sagemaker.inputs.TransformInput' instance.

`display_name` (str): The display name of the transform step.

description (str): The description of the transform step.

cache_config (CacheConfig): A ‘sagemaker.workflow.steps.CacheConfig‘ instance.

depends_on (List[str]): A list of step names this ‘sagemaker.workflow.steps.TransformStep‘ depends on.

retry_policies (List[RetryPolicy]): A list of retry policy

Method `to_request()`**:** Updates the dictionary with cache configuration.

Usage:

```
TransformStep$to_request()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TransformStep$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`transform_config` *Export Airflow transform config from a SageMaker transformer*

Description

Export Airflow transform config from a SageMaker transformer

Usage

```
transform_config(
  transformer,
  data,
  data_type = "S3Prefix",
  content_type = NULL,
  compression_type = NULL,
  split_type = NULL,
  job_name = NULL,
  input_filter = NULL,
  output_filter = NULL,
  join_source = NULL
)
```

Arguments

transformer	(sagemaker.transformer.Transformer): The SageMaker transformer to export Airflow config from.
data	(str): Input data location in S3.

`data_type` (str): What the S3 location defines (default: 'S3Prefix'). Valid values: * 'S3Prefix' - the S3 URI defines a key name prefix. All objects with this prefix will be used as inputs for the transform job. * 'ManifestFile' - the S3 URI points to a single manifest file listing each S3 object to use as an input for the transform job.

`content_type` (str): MIME type of the input data (default: None).

`compression_type` (str): Compression type of the input data, if compressed (default: None). Valid values: 'Gzip', None.

`split_type` (str): The record delimiter for the input object (default: 'None'). Valid values: 'None', 'Line', 'RecordIO', and 'TFRecord'.

`job_name` (str): job name (default: None). If not specified, one will be generated.

`input_filter` (str): A JSONPath to select a portion of the input to pass to the algorithm container for inference. If you omit the field, it gets the value '$', representing the entire input. For CSV data, each row is taken as a JSON array, so only index-based JSONPaths can be applied, e.g. $[0], $[1:]. CSV data should follow the 'RFC format <https://tools.ietf.org/html/rfc4180>'_. See 'Supported JSONPath Operators <https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform-data-processing.html#data-processing-operators>'_ for a table of supported JSONPath operators. For more information, see the SageMaker API documentation for 'CreateTransformJob <https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTransformJob.h
Some examples: "$[1:]", "$.features" (default: None).

`output_filter` (str): A JSONPath to select a portion of the joined/original output to return as the output. For more information, see the SageMaker API documentation for 'CreateTransformJob <https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTransformJob.html>'_.
Some examples: "$[1:]", "$.prediction" (default: None).

`join_source` (str): The source of data to be joined to the transform output. It can be set to 'Input' meaning the entire input record will be joined to the inference result. You can use OutputFilter to select the useful portion before uploading to S3. (default: None). Valid values: Input, None.

Value

dict: Transform config that can be directly used by SageMakerTransformOperator in Airflow.

`transform_config_from_estimator` *Export Airflow transform config from a SageMaker estimator*

Description

Export Airflow transform config from a SageMaker estimator

Usage

```
transform_config_from_estimator(
  estimator,
  task_id,
  task_type,
  instance_count,
  instance_type,
  data,
  data_type = "S3Prefix",
  content_type = NULL,
  compression_type = NULL,
  split_type = NULL,
  job_name = NULL,
  model_name = NULL,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  env = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  role = NULL,
  volume_kms_key = NULL,
  model_server_workers = NULL,
  image_uri = NULL,
  vpc_config_override = NULL,
  input_filter = NULL,
  output_filter = NULL,
  join_source = NULL
)
```

Arguments

estimator	(sagemaker.model.EstimatorBase): The SageMaker estimator to export Airflow config from. It has to be an estimator associated with a training job.
task_id	(str): The task id of any airflow.contrib.operators.SageMakerTrainingOperator or airflow.contrib.operators.SageMakerTuningOperator that generates training jobs in the DAG. The transform config is built based on the training job generated in this operator.
task_type	(str): Whether the task is from SageMakerTrainingOperator or SageMakerTuningOperator. Values can be 'training', 'tuning' or None (which means training job is not from any task).
instance_count	(int): Number of EC2 instances to use.
instance_type	(str): Type of EC2 instance to use, for example, 'ml.c4.xlarge'.
data	(str): Input data location in S3.

data_type (str): What the S3 location defines (default: 'S3Prefix'). Valid values: * 'S3Prefix' - the S3 URI defines a key name prefix. All objects with this prefix will be used as inputs for the transform job. * 'ManifestFile' - the S3 URI points to a single manifest file listing each S3 object to use as an input for the transform job.

content_type (str): MIME type of the input data (default: None).

compression_type (str): Compression type of the input data, if compressed (default: None). Valid values: 'Gzip', None.

split_type (str): The record delimiter for the input object (default: 'None'). Valid values: 'None', 'Line', 'RecordIO', and 'TFRecord'.

job_name (str): transform job name (default: None). If not specified, one will be generated.

model_name (str): model name (default: None). If not specified, one will be generated.

strategy (str): The strategy used to decide how to batch records in a single request (default: None). Valid values: 'MultiRecord' and 'SingleRecord'.

assemble_with (str): How the output is assembled (default: None). Valid values: 'Line' or 'None'.

output_path (str): S3 location for saving the transform result. If not specified, results are stored to a default bucket.

output_kms_key (str): Optional. KMS key ID for encrypting the transform output (default: None).

accept (str): The accept header passed by the client to the inference endpoint. If it is supported by the endpoint, it will be the format of the batch transform output.

env (dict): Environment variables to be set for use during the transform job (default: None).

max_concurrent_transforms (int): The maximum number of HTTP requests to be made to each individual transform container at one time.

max_payload (int): Maximum size of the payload in a single HTTP request to the container in MB.

tags (list[dict]): List of tags for labeling a transform job. If none specified, then the tags used for the training job are used for the transform job.

role (str): The "ExecutionRoleArn" IAM Role ARN for the "Model", which is also used during transform jobs. If not specified, the role from the Estimator will be used.

volume_kms_key (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

model_server_workers (int): Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.

image_uri (str): A Docker image URI to use for deploying the model

vpc_config_override (dict[str, list[str]]): Override for VpcConfig set on the model. Default: use subnets and security groups from this Estimator. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

input_filter (str): A JSONPath to select a portion of the input to pass to the algorithm container for inference. If you omit the field, it gets the value '$', representing the entire input. For CSV data, each row is taken as a JSON array, so only index-based JSONPaths can be applied, e.g. $[0], $[1:]. CSV data should follow the 'RFC format <https://tools.ietf.org/html/rfc4180>'_. See 'Supported JSONPath Operators <https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform-data-processing.html#data-processing-operators>'_ for a table of supported JSONPath operators. For more information, see the SageMaker API documentation for 'CreateTransformJob <https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTransformJob.h
Some examples: "$[1:]", "$.features" (default: None).

output_filter (str): A JSONPath to select a portion of the joined/original output to return as the output. For more information, see the SageMaker API documentation for 'CreateTransformJob <https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTransformJob.html>'_.
Some examples: "$[1:]", "$.prediction" (default: None).

join_source (str): The source of data to be joined to the transform output. It can be set to 'Input' meaning the entire input record will be joined to the inference result. You can use OutputFilter to select the useful portion before uploading to S3. (default: None). Valid values: Input, None.

Value

dict: Transform config that can be directly used by SageMakerTransformOperator in Airflow.

TuningStep *Workflow TuningStep class*

Description

Tuning step for workflow.

Super classes

sagemaker.workflow::Entity -> sagemaker.workflow::Step -> sagemaker.workflow::ConfigurableRetryStep -> TuningStep

Active bindings

arguments The arguments dict that is used to call 'create_hyper_parameter_tuning_job'. NOTE: The CreateHyperParameterTuningJob request is not quite the args list that workflow needs. The HyperParameterTuningJobName attribute cannot be included.

properties A Properties object representing 'DescribeHyperParameterTuningJobResponse' and 'ListTrainingJobsForHyperParameterTuningJobResponse' data model.

Methods

Public methods:

- TuningStep$new()
- TuningStep$to_request()
- TuningStep$get_top_model_s3_uri()
- TuningStep$clone()

Method `new()`**:** Construct a TuningStep, given a 'HyperparameterTuner' instance. In addition to the tuner instance, the other arguments are those that are supplied to the 'fit' method of the 'sagemaker.tuner.HyperparameterTuner'.

Usage:

```
TuningStep$new(
  name,
  tuner,
  display_name = NULL,
  description = NULL,
  inputs = NULL,
  job_arguments = NULL,
  cache_config = NULL,
  depends_on = NULL,
  retry_policies = NULL
)
```

Arguments:

`name` (str): The name of the tuning step.

`tuner` (HyperparameterTuner): A 'sagemaker.tuner.HyperparameterTuner' instance.

`display_name` (str): The display name of the tuning step.

`description` (str): The description of the tuning step.

`inputs` : Information about the training data. Please refer to the "fit()" method of the associated estimator, as this can take any of the following forms:

- (str) - The S3 location where training data is saved.
- (dict[str, str] or dict[str, sagemaker.inputs.TrainingInput]) - If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:'~sagemaker.inputs.TrainingInput' objects.
- (sagemaker.inputs.TrainingInput) - Channel configuration for S3 data sources that can provide additional information about the training dataset. See :func:'sagemaker.inputs.TrainingInput' for full details.
- (sagemaker.session.FileSystemInput) - channel configuration for a file system data source that can provide additional information as well as the path to the training dataset.
- (sagemaker.amazon.amazon_estimator.RecordSet) - A collection of Amazon :class:~'Record' objects serialized and stored in S3. For use with an estimator for an Amazon algorithm.
- (sagemaker.amazon.amazon_estimator.FileSystemRecordSet) - Amazon SageMaker channel configuration for a file system data source for Amazon algorithms.
- (list[sagemaker.amazon.amazon_estimator.RecordSet]) - A list of :class:~'sagemaker.amazon.amazon_estimator.R objects, where each instance is a different channel of training data.

- (list[sagemaker.amazon.amazon_estimator.FileSystemRecordSet]) - A list of :class:~‘sagemaker.amazon.amazon_ objects, where each instance is a different channel of training data.

`job_arguments` (List[str]): A list of strings to be passed into the processing job. Defaults to ‘None‘.

`cache_config` (CacheConfig): A ‘sagemaker.workflow.steps.CacheConfig‘ instance.

`depends_on` (List[str] or List[Step]): A list of step names or step instance this ‘sagemaker.workflow.steps.ProcessingStep depends on

`retry_policies` (List[RetryPolicy]): A list of retry policy

Method `to_request()`**:** Updates the dictionary with cache configuration.

Usage:

```
TuningStep$to_request()
```

Method `get_top_model_s3_uri()`**:** Get the model artifact s3 uri from the top performing training jobs.

Usage:

```
TuningStep$get_top_model_s3_uri(top_k, s3_bucket, prefix = "")
```

Arguments:

`top_k` (int): the index of the top performing training job tuning step stores up to 50 top performing training jobs, hence a valid top_k value is from 0 to 49. The best training job model is at index 0

`s3_bucket` (str): the s3 bucket to store the training job output artifact

`prefix` (str): the s3 key prefix to store the training job output artifact

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TuningStep$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`tuning_config` *Export Airflow tuning config from a HyperparameterTuner*

Description

Export Airflow tuning config from a HyperparameterTuner

Usage

```
tuning_config(
  tuner,
  inputs,
  job_name = NULL,
  include_cls_metadata = FALSE,
  mini_batch_size = NULL
)
```

Arguments

tuner	(sagemaker.tuner.HyperparameterTuner): The tuner to export tuning config from.
inputs	: Information about the training data. Please refer to the "fit()" method of the associated estimator in the tuner, as this can take any of the following forms: * (str) - The S3 location where training data is saved. * (dict[str, str] or dict[str, sagemaker.inputs.TrainingInput]) - If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:'~sagemaker.inputs.TrainingInput' objects. * (sagemaker.inputs.TrainingInput) - Channel configuration for S3 data sources that can provide additional information about the training dataset. See :func:'sagemaker.inputs.TrainingInput' for full details. * (sagemaker.amazon.amazon_estimator.RecordS - A collection of Amazon :class:~'Record' objects serialized and stored in S3. For use with an estimator for an Amazon algorithm. * (list[sagemaker.amazon.amazon_estimator.RecordS - A list of :class:~'sagemaker.amazon.amazon_estimator.RecordSet' objects, where each instance is a different channel of training data. * (dict[str, one the forms above]): Required by only tuners created via the factory method "HyperparameterTuner.create()". The keys should be the same estimator names as keys for the "estimator_list" argument of the "HyperparameterTuner.create()" method.
job_name	(str): Specify a tuning job name if needed.
include_cls_metadata	: It can take one of the following two forms. * (bool) - Whether or not the hyperparameter tuning job should include information about the estimator class (default: False). This information is passed as a hyperparameter, so if the algorithm you are using cannot handle unknown hyperparameters (e.g. an Amazon SageMaker built-in algorithm that does not have a custom estimator in the Python SDK), then set "include_cls_metadata" to "False". * (dict[str, bool]) - This version should be used for tuners created via the factory method "HyperparameterTuner.create()", to specify the flag for individual estimators provided in the "estimator_list" argument of the method. The keys would be the same estimator names as in "estimator_list". If one estimator doesn't need the flag set, then no need to include it in the dictionary. If none of the estimators need the flag set, then an empty dictionary "" must be used.
mini_batch_size	: It can take one of the following two forms. * (int) - Specify this argument only when estimator is a built-in estimator of an Amazon algorithm. For other estimators, batch size should be specified in the estimator. * (dict[str, int]) - This version should be used for tuners created via the factory method "HyperparameterTuner.create()", to specify the value for individual estimators provided in the "estimator_list" argument of the method. The keys would be the same estimator names as in "estimator_list". If one estimator doesn't need the value set, then no need to include it in the dictionary. If none of the estimators need the value set, then an empty dictionary "" must be used.

Value

list: Tuning config that can be directly used by SageMakerTuningOperator in Airflow.

update_estimator_from_task *Update training job of the estimator from a task in the DAG*

Description

Update training job of the estimator from a task in the DAG

Usage

```
update_estimator_from_task(estimator, task_id, task_type)
```

Arguments

estimator	(sagemaker.estimator.EstimatorBase): The estimator to update
task_id	(str): The task id of any airflow.contrib.operators.SageMakerTrainingOperator or airflow.contrib.operators.SageMakerTuningOperator that generates training jobs in the DAG.
task_type	(str): Whether the task is from SageMakerTrainingOperator or SageMakerTuningOperator. Values can be 'training', 'tuning' or None (which means training job is not from any task).

update_submit_s3_uri *Updated the S3 URI of the framework source directory in given estimator.*

Description

Updated the S3 URI of the framework source directory in given estimator.

Usage

```
update_submit_s3_uri(estimator, job_name)
```

Arguments

estimator	(sagemaker.estimator.Framework): The Framework estimator to update.
job_name	(str): The new job name included in the submit S3 URI

Value

str: The updated S3 URI of framework source directory

Index