include::attributes.txt[]

// Attributes [.topic] [#how-to-set-cw-alarm] = Set a CloudWatch alarm :info\_titleabbrev: Set CloudWatch alarm

// Content start

Use the link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_cloudwatch-readme.html[aws-cloudwatch] package to set up Amazon CloudWatch alarms on CloudWatch metrics. You can use predefined metrics or create your own.

[#how-to-set-cw-alarm-use-metric] == Use an existing metric

Many {aws} Construct Library modules let you set an alarm on an existing metric by passing the metric’s name to a convenience method on an instance of an object that has metrics. For example, given an Amazon SQS queue, you can get the metric *ApproximateNumberOfMessagesVisible* from the queue’s link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_sqs.Queue.html#metricmetricname-props[metric()] method:

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const metric = queue.metric(“ApproximateNumberOfMessagesVisible”); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const metric = queue.metric(“ApproximateNumberOfMessagesVisible”); —

Python:: + [source,python,subs=“verbatim,attributes”] — metric = queue.metric(“ApproximateNumberOfMessagesVisible”) —

Java:: + [source,java,subs=“verbatim,attributes”] — Metric metric = queue.metric(“ApproximateNumberOfMessagesVisible”); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var metric = queue.Metric(“ApproximateNumberOfMessagesVisible”); — ====

[#how-to-set-cw-alarm-new-metric] == Create your own metric

Create your own link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_cloudwatch.Metric.html[metric] as follows, where the *namespace* value should be something like *{aws}/SQS* for an Amazon SQS queue. You also need to specify your metric’s name and dimension:

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const metric = new cloudwatch.Metric({ namespace: ‘MyNamespace’, metricName: ‘MyMetric’, dimensionsMap: { MyDimension: ‘MyDimensionValue’ } }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const metric = new cloudwatch.Metric({ namespace: ‘MyNamespace’, metricName: ‘MyMetric’, dimensionsMap: { MyDimension: ‘MyDimensionValue’ } }); —

Python:: + [source,python,subs=“verbatim,attributes”] — metric = cloudwatch.Metric( namespace=“MyNamespace”, metric\_name=“MyMetric”, dimensionsMap=dict(MyDimension=“MyDimensionValue”) ) —

Java:: + [source,java,subs=“verbatim,attributes”] — Metric metric = Metric.Builder.create() .namespace(“MyNamespace”) .metricName(“MyMetric”) .dimensionsMap(java.util.Map.of( // Java 9 or later “MyDimension”, “MyDimensionValue”)) .build(); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var metric = new Metric(this, “Metric”, new MetricProps { Namespace = “MyNamespace”, MetricName = “MyMetric”, Dimensions = new Dictionary<string, object> { { “MyDimension”, “MyDimensionValue” } } }); — ====

[#how-to-set-cw-alarm-create] == Create the alarm

Once you have a metric, either an existing one or one you defined, you can create an alarm. In this example, the alarm is raised when there are more than 100 of your metric in two of the last three evaluation periods. You can use comparisons such as less-than in your alarms via the comparisonOperator property. Greater-than-or-equal-to is the {aws} CDK default, so we don’t need to specify it.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const alarm = new cloudwatch.Alarm(this, ‘Alarm’, { metric: metric, threshold: 100, evaluationPeriods: 3, datapointsToAlarm: 2, }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const alarm = new cloudwatch.Alarm(this, ‘Alarm’, { metric: metric, threshold: 100, evaluationPeriods: 3, datapointsToAlarm: 2 }); —

Python:: + [source,python,subs=“verbatim,attributes”] — alarm = cloudwatch.Alarm(self, “Alarm”, metric=metric, threshold=100, evaluation\_periods=3, datapoints\_to\_alarm=2 ) —

Java:: + [source,java,subs=“verbatim,attributes”] — import software.amazon.awscdk.services.cloudwatch.Alarm; import software.amazon.awscdk.services.cloudwatch.Metric;

Alarm alarm = Alarm.Builder.create(this, “Alarm”) .metric(metric) .threshold(100) .evaluationPeriods(3) .datapointsToAlarm(2).build(); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var alarm = new Alarm(this, “Alarm”, new AlarmProps { Metric = metric, Threshold = 100, EvaluationPeriods = 3, DatapointsToAlarm = 2 }); — ====

An alternative way to create an alarm is using the metric’s link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_cloudwatch.Metric.html#createwbralarmscope-id-props[createAlarm()] method, which takes essentially the same properties as the Alarm constructor. You don’t need to pass in the metric, because it’s already known.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — metric.createAlarm(this, ‘Alarm’, { threshold: 100, evaluationPeriods: 3, datapointsToAlarm: 2, }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — metric.createAlarm(this, ‘Alarm’, { threshold: 100, evaluationPeriods: 3, datapointsToAlarm: 2, }); —

Python:: + [source,python,subs=“verbatim,attributes”] — metric.create\_alarm(self, “Alarm”, threshold=100, evaluation\_periods=3, datapoints\_to\_alarm=2 ) —

Java:: + [source,java,subs=“verbatim,attributes”] — metric.createAlarm(this, “Alarm”, new CreateAlarmOptions.Builder() .threshold(100) .evaluationPeriods(3) .datapointsToAlarm(2) .build()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — metric.CreateAlarm(this, “Alarm”, new CreateAlarmOptions { Threshold = 100, EvaluationPeriods = 3, DatapointsToAlarm = 2 }); — ====