Amazon CloudWatch

User Guide

aws

Amazon CloudWatch: User Guide

Table of Contents

What is Amazon CloudWatch?

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use this data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money. In addition to monitoring the built-in metrics that come with AWS, you can monitor your own custom metrics. With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.

Accessing CloudWatch

You can access CloudWatch using any of the following methods:

- **Amazon CloudWatch console** — https://console.aws.amazon.com/cloudwatch/
- **AWS CLI** — For more information, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
- **CloudWatch API** — For more information, see the Amazon CloudWatch API Reference.
- **AWS SDKs** — For more information, see Tools for Amazon Web Services.

Related AWS Services

The following services are used along with Amazon CloudWatch:

- **Amazon Simple Notification Service (Amazon SNS)** coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. You use Amazon SNS with CloudWatch to send messages when an alarm threshold has been reached. For more information, see Set Up Amazon SNS Notifications (p. 331).
- **Amazon EC2 Auto Scaling** enables you to automatically launch or terminate Amazon EC2 instances based on user-defined policies, health status checks, and schedules. You can use a CloudWatch alarm with Amazon EC2 Auto Scaling to scale your EC2 instances based on demand. For more information, see Dynamic Scaling in the *Amazon EC2 Auto Scaling User Guide*.
- **AWS CloudTrail** enables you to monitor the calls made to the Amazon CloudWatch API for your account, including calls made by the AWS Management Console, AWS CLI, and other services. When CloudTrail logging is turned on, CloudWatch writes log files to the Amazon S3 bucket that you specified when you configured CloudTrail. For more information, see Logging Amazon CloudWatch API Calls with AWS CloudTrail (p. 371).
- **AWS Identity and Access Management (IAM)** is a web service that helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication) and what resources they can use in which ways (authorization). For more information, see Authentication and Access Control for Amazon CloudWatch (p. 351).

How Amazon CloudWatch Works

Amazon CloudWatch is basically a metrics repository. An AWS service—such as Amazon EC2—puts metrics into the repository, and you retrieve statistics based on those metrics. If you put your own custom metrics into the repository, you can retrieve statistics on these metrics as well.
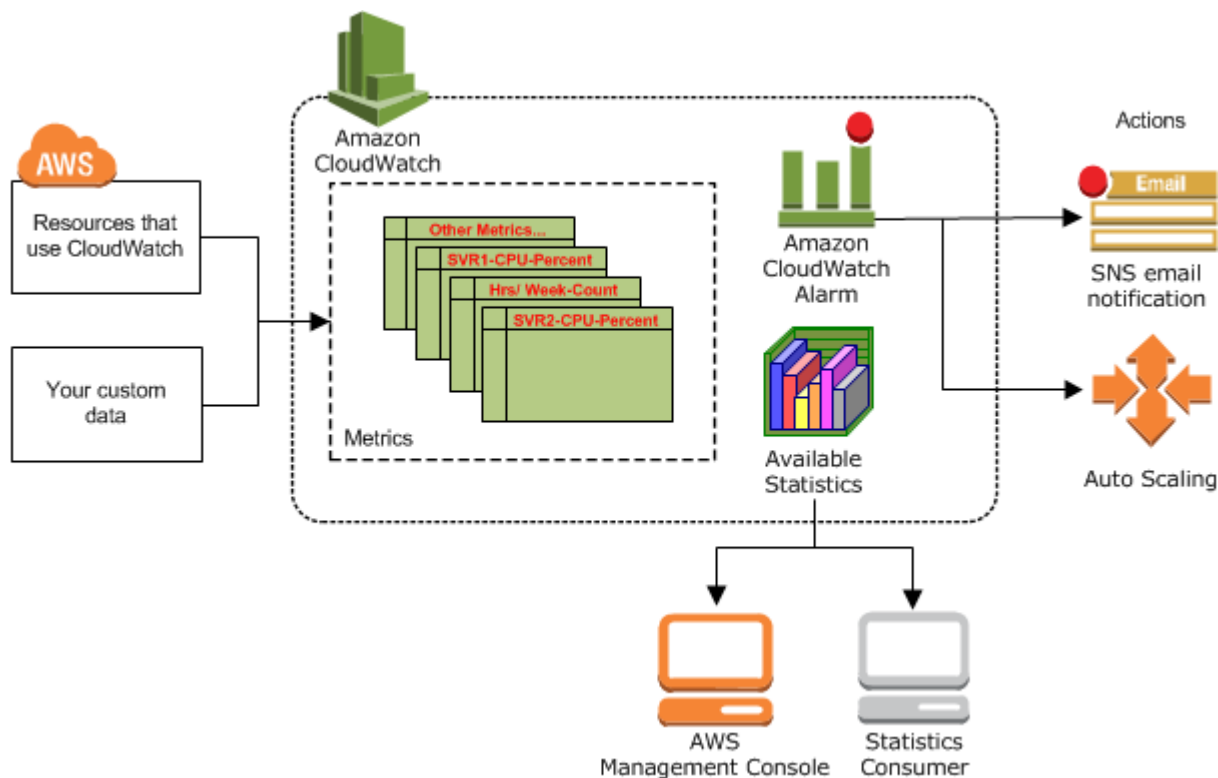
You can use metrics to calculate statistics and then present the data graphically in the CloudWatch console. For more information about the other AWS resources that generate and send metrics to CloudWatch, see Amazon CloudWatch Metrics and Dimensions Reference (p. 114).

You can configure alarm actions to stop, start, or terminate an Amazon EC2 instance when certain criteria are met. In addition, you can create alarms that initiate Amazon EC2 Auto Scaling and Amazon Simple Notification Service (Amazon SNS) actions on your behalf. For more information about creating CloudWatch alarms, see Alarms (p. 7).

AWS Cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, each data center facility is located in a specific geographical area, known as a *region*. Each region is designed to be completely isolated from the other regions, to achieve the greatest possible failure isolation and stability. Amazon CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*.

Amazon CloudWatch Concepts

The following terminology and concepts are central to your understanding and use of Amazon CloudWatch:

- Namespaces (p. 3)
- Metrics (p. 3)
- Dimensions (p. 4)
- Statistics (p. 5)
- Percentiles (p. 7)
- Alarms (p. 7)

Namespaces

A *namespace* is a container for CloudWatch metrics. Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.

There is no default namespace. You must specify a namespace for each data point you publish to CloudWatch. You can specify a namespace name when you create a metric. These names must contain valid XML characters, and be fewer than 256 characters in length. Possible characters are: alphanumeric characters (0-9A-Za-z), period (.), hyphen (-), underscore (_), forward slash (/), hash (#), and colon (:).

The AWS namespaces use the following naming convention: `AWS/service`. For example, Amazon EC2 uses the `AWS/EC2` namespace. For the list of AWS namespaces, see AWS Namespaces (p. 115).

Metrics

Metrics are the fundamental concept in CloudWatch. A metric represents a time-ordered set of data points that are published to CloudWatch. Think of a metric as a variable to monitor, and the data points represent the values of that variable over time. For example, the CPU usage of a particular EC2 instance is one metric provided by Amazon EC2. The data points themselves can come from any application or business activity from which you collect data.

AWS services send metrics to CloudWatch, and you can send your own custom metrics to CloudWatch. You can add the data points in any order, and at any rate you choose. You can retrieve statistics about those data points as an ordered set of time-series data.

Metrics exist only in the region in which they are created. Metrics cannot be deleted, but they automatically expire after 15 months if no new data is published to them. Data points older than 15 months expire on a rolling basis; as new data points come in, data older than 15 months is dropped.

Metrics are uniquely defined by a name, a namespace, and zero or more dimensions. Each data point has a time stamp, and (optionally) a unit of measure. When you request statistics, the returned data stream is identified by namespace, metric name, dimension, and (optionally) the unit.

For more information, see View Available Metrics (p. 29) and Publish Custom Metrics (p. 45).

Time Stamps

Each metric data point must be marked with a time stamp. The time stamp can be up to two weeks in the past and up to two hours into the future. If you do not provide a time stamp, CloudWatch creates a time stamp for you based on the time the data point was received.

Time stamps are `dateTime` objects, with the complete date plus hours, minutes, and seconds (for example, 2016-10-31T23:59:59Z). For more information, see dateTime. Although it is not required, we recommend that you use Coordinated Universal Time (UTC). When you retrieve statistics from CloudWatch, all times are in UTC.

CloudWatch alarms check metrics based on the current time in UTC. Custom metrics sent to CloudWatch with time stamps other than the current UTC time can cause alarms to display the **Insufficient Data** state or result in delayed alarms.

Metrics Retention

CloudWatch retains metric data as follows:

- Data points with a period of less than 60 seconds are available for 3 hours. These data points are high-resolution custom metrics.
- Data points with a period of 60 seconds (1 minute) are available for 15 days

- Data points with a period of 300 seconds (5 minute) are available for 63 days
- Data points with a period of 3600 seconds (1 hour) are available for 455 days (15 months)

Data points that are initially published with a shorter period are aggregated together for long-term storage. For example, if you collect data using a period of 1 minute, the data remains available for 15 days with 1-minute resolution. After 15 days this data is still available, but is aggregated and is retrievable only with a resolution of 5 minutes. After 63 days, the data is further aggregated and is available with a resolution of 1 hour.

CloudWatch started retaining 5-minute and 1-hour metric data as of 9 July 2016.

Dimensions

A *dimension* is a name/value pair that uniquely identifies a metric. You can assign up to 10 dimensions to a metric.

Every metric has specific characteristics that describe it, and you can think of dimensions as categories for those characteristics. Dimensions help you design a structure for your statistics plan. Because dimensions are part of the unique identifier for a metric, whenever you add a unique name/value pair to one of your metrics, you are creating a new variation of that metric.

AWS services that send data to CloudWatch attach dimensions to each metric. You can use dimensions to filter the results that CloudWatch returns. For example, you can get statistics for a specific EC2 instance by specifying the `InstanceId` dimension when you search for metrics.

For metrics produced by certain AWS services, such as Amazon EC2, CloudWatch can aggregate data across dimensions. For example, if you search for metrics in the `AWS/EC2` namespace but do not specify any dimensions, CloudWatch aggregates all data for the specified metric to create the statistic that you requested. CloudWatch does not aggregate across dimensions for your custom metrics.

Dimension Combinations

CloudWatch treats each unique combination of dimensions as a separate metric, even if the metrics have the same metric name. You can only retrieve statistics using combinations of dimensions that you specifically published. When you retrieve statistics, specify the same values for the namespace, metric name, and dimension parameters that were used when the metrics were created. You can also specify the start and end times for CloudWatch to use for aggregation.

For example, suppose that you publish four distinct metrics named ServerStats in the DataCenterMetric namespace with the following properties:

```
Dimensions: Server=Prod, Domain=Frankfurt, Unit: Count, Timestamp: 2016-10-31T12:30:00Z,
 Value: 105
Dimensions: Server=Beta, Domain=Frankfurt, Unit: Count, Timestamp: 2016-10-31T12:31:00Z,
 Value: 115
Dimensions: Server=Prod, Domain=Rio,       Unit: Count, Timestamp: 2016-10-31T12:32:00Z,
 Value: 95
Dimensions: Server=Beta, Domain=Rio,       Unit: Count, Timestamp: 2016-10-31T12:33:00Z,
 Value: 97
```

If you publish only those four metrics, you can retrieve statistics for these combinations of dimensions:

- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`
- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

You can't retrieve statistics for the following dimensions or if you specify no dimensions:

- `Server=Prod`
- `Server=Beta`
- `Domain=Frankfurt`
- `Domain=Rio`

Statistics

Statistics are metric data aggregations over specified periods of time. CloudWatch provides statistics based on the metric data points provided by your custom data or provided by other AWS services to CloudWatch. Aggregations are made using the namespace, metric name, dimensions, and the data point unit of measure, within the time period you specify. The following table describes the available statistics.

Statistic	Description
`Minimum`	The lowest value observed during the specified period. You can use this value to determine low volumes of activity for your application.
`Maximum`	The highest value observed during the specified period. You can use this value to determine high volumes of activity for your application.
`Sum`	All values submitted for the matching metric added together. This statistic can be useful for determining the total volume of a metric.
`Average`	The value of `Sum` / `SampleCount` during the specified period. By comparing this statistic with the `Minimum` and `Maximum`, you can determine the full scope of a metric and how close the average use is to the `Minimum` and `Maximum`. This comparison helps you to know when to increase or decrease your resources as needed.
`SampleCount`	The count (number) of data points used for the statistical calculation.
`pNN.NN`	The value of the specified percentile. You can specify any percentile, using up to two decimal places (for example, p95.45). Percentile statistics are not available for metrics that include any negative values. For more information, see Percentiles (p. 7).

You can add pre-calculated statistics. Instead of data point values, you specify values for `SampleCount`, `Minimum`, `Maximum`, and `Sum` (CloudWatch calculates the average for you). The values you add in this way are aggregated with any other values associated with the matching metric.

Units

Each statistic has a unit of measure. Example units include `Bytes`, `Seconds`, `Count`, and `Percent`. For the complete list of the units that CloudWatch supports, see the MetricDatum data type in the *Amazon CloudWatch API Reference*.

You can specify a unit when you create a custom metric. If you do not specify a unit, CloudWatch uses `None` as the unit. Units help provide conceptual meaning to your data. Though CloudWatch attaches no significance to a unit internally, other applications can derive semantic information based on the unit.

Metric data points that specify a unit of measure are aggregated separately. When you get statistics without specifying a unit, CloudWatch aggregates all data points of the same unit together. If you have two otherwise identical metrics with different units, two separate data streams are returned, one for each unit.

Periods

A *period* is the length of time associated with a specific Amazon CloudWatch statistic. Each statistic represents an aggregation of the metrics data collected for a specified period of time. Periods are defined in numbers of seconds, and valid values for period are 1, 5, 10, 30, or any multiple of 60. For example, to specify a period of six minutes, use 360 as the period value. You can adjust how the data is aggregated by varying the length of the period. A period can be as short as one second or as long as one day (86,400 seconds). The default value is 60 seconds.

Only custom metrics that you define with a storage resolution of 1 second support sub-minute periods. Even though the option to set a period below 60 is always available in the console, you should select a period that aligns to how the metric is stored. For more information about metrics that support sub-minute periods, see High-Resolution Metrics (p. 45).

When you retrieve statistics, you can specify a period, start time, and end time. These parameters determine the overall length of time associated with the statistics. The default values for the start time and end time get you the last hour's worth of statistics. The values that you specify for the start time and end time determine how many periods CloudWatch returns. For example, retrieving statistics using the default values for the period, start time, and end time returns an aggregated set of statistics for each minute of the previous hour. If you prefer statistics aggregated in ten-minute blocks, specify a period of 600. For statistics aggregated over the entire hour, specify a period of 3600.

When statistics are aggregated over a period of time, they are stamped with the time corresponding to the beginning of the period. For example, data aggregated from 7:00pm to 8:00pm is stamped as 7:00pm. Additionally, data aggregated between 7:00pm and 8:00pm begins to be visible at 7:00pm, then the values of that aggregated data may change as CloudWatch collects more samples during the period.

Periods are also important for CloudWatch alarms. When you create an alarm to monitor a specific metric, you are asking CloudWatch to compare that metric to the threshold value that you specified. You have extensive control over how CloudWatch makes that comparison. Not only can you specify the period over which the comparison is made, but you can also specify how many evaluation periods are used to arrive at a conclusion. For example, if you specify three evaluation periods, CloudWatch compares a window of three data points. CloudWatch only notifies you if the oldest data point is breaching and the others are breaching or missing. For metrics that are continuously emitted, CloudWatch doesn't notify you until three failures are found.

Aggregation

Amazon CloudWatch aggregates statistics according to the period length that you specify when retrieving statistics. You can publish as many data points as you want with the same or similar time stamps. CloudWatch aggregates them by period length. Aggregated statistics are only available when using detailed monitoring. In addition, Amazon CloudWatch does not aggregate data across regions.

You can publish data points for a metric that share not only the same time stamp, but also the same namespace and dimensions. CloudWatch returns aggregated statistics for those data points. You can also publish multiple data points for the same or different metrics, with any time stamp.

For large datasets, you can insert a pre-aggregated dataset called a *statistic set*. With statistic sets, you give CloudWatch the Min, Max, Sum, and SampleCount for a number of data points. This is commonly used when you need to collect data many times in a minute. For example, suppose you have a metric for the request latency of a webpage. It doesn't make sense to publish data with every webpage hit. We suggest that you collect the latency of all hits to that webpage, aggregate them once a minute, and send that statistic set to CloudWatch.

Amazon CloudWatch doesn't differentiate the source of a metric. If you publish a metric with the same namespace and dimensions from different sources, CloudWatch treats this as a single metric. This can be useful for service metrics in a distributed, scaled system. For example, all the hosts in a web server application could publish identical metrics representing the latency of requests they are processing.

CloudWatch treats these as a single metric, allowing you to get the statistics for minimum, maximum, average, and sum of all requests across your application.

Percentiles

A *percentile* indicates the relative standing of a value in a dataset. For example, the 95th percentile means that 95 percent of the data is lower than this value and 5 percent of the data is higher than this value. Percentiles help you get a better understanding of the distribution of your metric data. You can use percentiles with the following services:

- Amazon EC2
- Amazon RDS
- Kinesis
- Application Load Balancer
- Elastic Load Balancing
- API Gateway

Percentiles are often used to isolate anomalies. In a typical distribution, 95 percent of the data is within two standard deviations from the mean and 99.7 percent of the data is within three standard deviations from the mean. Any data that falls outside three standard deviations is often considered to be an anomaly because it differs so greatly from the average value. For example, suppose that you are monitoring the CPU utilization of your EC2 instances to ensure that your customers have a good experience. If you monitor the average, this can hide anomalies. If you monitor the maximum, a single anomaly can skew the results. Using percentiles, you can monitor the 95th percentile of CPU utilization to check for instances with an unusually heavy load.

You can monitor your system and applications using percentiles as you would use the other CloudWatch statistics (Average, Minimum, Maximum, and Sum). For example, when you create an alarm, you can use percentiles as the statistical function. You can specify the percentile with up to two decimal places (for example, p95.45).

Percentile statistics are available for custom metrics as well as metrics from AWS services, as long as you publish the raw, unsummarized data points for your custom metric. Percentile statistics are not available for metrics when any of the metric values are negative numbers.

CloudWatch needs raw data points to calculate percentiles. If you publish data using a statistic set instead, you can only retrieve percentile statistics for this data when one of the following conditions is true:

- The SampleCount of the statistic set is 1.
- The Min and the Max of the statistic set are equal.

Alarms

You can use an *alarm* to automatically initiate actions on your behalf. An alarm watches a single metric over a specified time period, and performs one or more specified actions, based on the value of the metric relative to a threshold over time. The action is a notification sent to an Amazon SNS topic or an Auto Scaling policy. You can also add alarms to dashboards.

Alarms invoke actions for sustained state changes only. CloudWatch alarms do not invoke actions simply because they are in a particular state. The state must have changed and been maintained for a specified number of periods.

When creating an alarm, select a period that is greater than or equal to the frequency of the metric to be monitored. For example, basic monitoring for Amazon EC2 provides metrics for your instances every 5

minutes. When setting an alarm on a basic monitoring metric, select a period of at least 300 seconds (5 minutes). Detailed monitoring for Amazon EC2 provides metrics for your instances every 1 minute. When setting an alarm on a detailed monitoring metric, select a period of at least 60 seconds (1 minute).

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. There is a higher charge for high-resolution alarms. For more information about high-resolution metrics, see Publish Custom Metrics (p. 45).

For more information, see Creating Amazon CloudWatch Alarms (p. 327) and Create an Alarm from a Metric on a Graph (p. 44).

CloudWatch Limits

CloudWatch has the following limits:

Resource	Default Limit
Actions	5/alarm. This limit cannot be changed.
Alarms	10/month/customer for free. 5000 per region per account.
API requests	1,000,000/month/customer for free.
Custom metrics	No limit.
Dashboards	Up to 1000 dashboards per account. Up to 100 metrics per dashboard widget. Up to 500 metrics per dashboard, across all widgets. These limits cannot be changed.
DescribeAlarms	9 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a limit increase.
Dimensions	10/metric. This limit cannot be changed.
GetMetricData	50 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. 180,000 Datapoints Per Second (DPS) if the `StartTime` used in the API request is less than or equal to three hours from current time. 90,000 DPS if the `StartTime` is more than three hours from current time. This is the maximum number of datapoints you can request per second using one or more API calls without being throttled. You can request a limit increase for both of these limits.
GetMetricData	A single `GetMetricData` call can include as many as 100 `MetricDataQuery` structures.

Resource	Default Limit
	This limit cannot be changed.
GetMetricStatistics	400 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a limit increase.
ListMetrics	25 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a limit increase.
Metric data	15 months. This limit cannot be changed.
MetricDatum items	20/PutMetricData request. A MetricDatum object can contain a single value or a StatisticSet object representing many values. This limit cannot be changed.
Metrics	10/month/customer for free.
Period	Maximum value is one day (86,400 seconds). This limit cannot be changed.
PutMetricAlarm request	3 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a limit increase.
PutMetricData request	40 KB for HTTP POST requests. PutMetricData can handle 150 transactions per second (TPS), which is the maximum number of operation requests you can make per second without being throttled. You can request a limit increase.
Amazon SNS email notifications	1,000/month/customer for free.

Amazon CloudWatch Resources

The following related resources can help you as you work with this service.

Resource	Description
Amazon CloudWatch FAQs	The FAQ covers the top questions developers have asked about this product.
Release notes	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code examples, release notes, and other information to help you build innovative applications with AWS.

Resource	Description
AWS Management Console	The console allows you to perform most of the functions of Amazon CloudWatch and various other AWS offerings without programming.
Amazon CloudWatch Discussion Forums	Community-based forum for developers to discuss technical questions related to Amazon CloudWatch.
AWS Support	The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
Amazon CloudWatch product information	The primary webpage for information about Amazon CloudWatch.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse, etc.

Getting Set Up

To use Amazon CloudWatch you need an AWS account. Your AWS account allows you to use services (for example, Amazon EC2) to generate metrics that you can view in the CloudWatch console, a point-and-click web-based interface. In addition, you can install and configure the AWS command line interface (CLI).

Sign Up for Amazon Web Services (AWS)

When you create an AWS account, we automatically sign up your account for all AWS services. You pay only for the services that you use.

If you have an AWS account already, skip to the next step. If you don't have an AWS account, use the following procedure to create one.

To sign up for an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.

   Note
   If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code using the phone keypad.

Sign in to the Amazon CloudWatch Console

To sign in to the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, use the navigation bar to change the region to the region where you have your AWS resources.
3. Even if this is the first time you are using the CloudWatch console, **Your Metrics** could already report metrics, because you have used a AWS product that automatically pushes metrics to Amazon CloudWatch for free. Other AWS products require that you enable metrics.

   If you do not have any alarms, the **Your Alarms** section will have a **Create Alarm** button.

Set Up the AWS CLI

You can use the AWS CLI or the Amazon CloudWatch CLI to perform CloudWatch commands. Note that the AWS CLI replaces the CloudWatch CLI; we include new CloudWatch features only in the AWS CLI.

For information about how to install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

For information about how to install and configure the Amazon CloudWatch CLI, see Set Up the Command Line Interface in the *Amazon CloudWatch CLI Reference*.

Getting Started with Amazon CloudWatch

The following scenarios show you how to use Amazon CloudWatch. In the first scenario, you use the CloudWatch console to create a billing alarm that tracks your AWS usage and lets you know when you have exceeded a certain spending threshold. In the second, more advanced scenario, you use the AWS command line interface (CLI) to publish a single metric for a hypothetical application named *GetStarted*.

Scenarios

Scenario: Monitor Your Estimated Charges Using CloudWatch

In this scenario, you create an Amazon CloudWatch alarm to monitor your estimated charges. When you enable the monitoring of estimated charges for your AWS account, the estimated charges are calculated and sent several times daily to CloudWatch as metric data.

Billing metric data is stored in the US East (N. Virginia) Region and reflects worldwide charges. This data includes the estimated charges for every service in AWS that you use, as well as the estimated overall total of your AWS charges.

You can choose to receive alerts by email when charges have exceeded a certain threshold. These alerts are triggered by CloudWatch and messages are sent using Amazon Simple Notification Service (Amazon SNS).

Tasks

Step 1: Enable Billing Alerts

Before you can create an alarm for your estimated charges, you must enable billing alerts, so that you can monitor your estimated AWS charges and create an alarm using billing metric data. After you enable billing alerts, you cannot disable data collection, but you can delete any billing alarms you created.

After you enable billing alerts for the first time, it takes about 15 minutes before you can view billing data and set billing alarms.

Requirements

- You must be signed in using root account credentials; IAM users cannot enable billing alerts for your AWS account.
- For consolidated billing accounts, billing data for each linked account can be found by logging in as the paying account. You can view billing data for total estimated charges and estimated charges by service for each linked account as well as for the consolidated account.

To enable monitoring of your estimated charges

1. Open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home?#.
2. In the navigation pane, choose **Preferences**.
3. Select **Receive Billing Alerts**.

4. Choose **Save preferences**.

Step 2: Create a Billing Alarm

After you've enabled billing alerts, you can create a billing alarm. In this scenario, you create an alarm that sends an email message when your estimated charges for AWS exceed a specified threshold.

> **Note**
> This procedure uses the simple options. To use the advanced options, see Create a Billing Alarm (p. 347) in *Create a Billing Alarm to Monitor Your Estimated AWS Charges*.

To create a billing alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**, **Billing**.
4. For **Whenever my total AWS charges for the month exceed**, specify the monetary amount (for example, 200) that must be exceeded to trigger the alarm and send an email notification.

> **Tip**
> Under **Alarm Preview**, there is an estimate of your charges that you can use to set an appropriate amount.

5. For **send a notification to**, choose an existing notification list or create a new one.

   To create a list, choose **New list** and type a comma-separated list of email addresses to be notified when the alarm changes to the ALARM state. Each email address will be sent a subscription confirmation email. The recipient must confirm the subscription before notifications can be sent to the email address.

6. Choose **Create Alarm**.

Step 3: Check the Alarm Status

Now, check the status of the billing alarm that you just created.

To check the alarm status

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**, **Billing**.
4. Select the check box next to the alarm. Note that until the subscription is confirmed, it is shown as "Pending confirmation". After the subscription is confirmed, refresh the console to show the updated status.

Step 4: Edit a Billing Alarm

Let's say that you want to increase the amount money you spend with AWS each month from $200 to $400. You can edit your existing billing alarm and increase the monetary amount that must be exceeded before the alarm is triggered.

To edit a billing alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and reflects worldwide charges.

3. In the navigation pane, choose **Alarms**, **Billing**.

4. Select the check box next to the alarm and then choose **Actions**, **Modify**.

5. For **Whenever my total AWS charges for the month exceed**, specify the new amount that must be exceeded to trigger the alarm and send an email notification.

6. Choose **Save Changes**.

Step 5: Delete a Billing Alarm

You can delete your billing alarm if you no longer need it.

To delete a billing alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and reflects worldwide charges.

3. In the navigation pane, choose **Alarms**, **Billing**.

4. Select the check box next to the alarm and then choose **Actions**, **Delete**.

5. When prompted for confirmation, choose **Yes, Delete**.

Scenario: Publish Metrics to CloudWatch

In this scenario, you'll use the AWS Command Line Interface (AWS CLI) to publish a single metric for a hypothetical application named *GetStarted*. If you haven't already installed and configured the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

Tasks

- Step 1: Define the Data Configuration (p. 16)
- Step 2: Add Metrics to CloudWatch (p. 17)
- Step 3: Get Statistics from CloudWatch (p. 18)
- Step 4: View Graphs with the Console (p. 18)

Step 1: Define the Data Configuration

In this scenario, you'll publish data points that track the request latency for the application. Choose names for your metric and namespace that make sense to you. For this example, name the metric *RequestLatency* and place all of the data points into the *GetStarted* namespace.

You'll publish several data points that collectively represent three hours of latency data. The raw data comprises fifteen request latency readings distributed over three hours. Each reading is in milliseconds:

- Hour one: 87, 51, 125, 235

- Hour two: 121, 113, 189, 65, 89
- Hour three: 100, 47, 133, 98, 100, 328

You can publish data to CloudWatch as single data points or as an aggregated set of data points called a *statistic set*. You can aggregate metrics to a granularity as low as one minute. You can publish the aggregated data points to CloudWatch as a set of statistics with four predefined keys: `Sum`, `Minimum`, `Maximum`, and `SampleCount`.

You'll publish the data points from hour one as single data points. For the data from hours two and three, you'll aggregate the data points and publish a statistic set for each hour. The key values are shown in the following table.

Hour	Raw Data	Sum	Minimum	Maximum	SampleCount
1	87				
1	51				
1	125				
1	235				
2	121, 113, 189, 65, 89	577	65	189	5
3	100, 47, 133, 98, 100, 328	806	47	328	6

Step 2: Add Metrics to CloudWatch

After you have defined your data configuration, you are ready to add data.

To publish data points to CloudWatch

1. At a command prompt, run the following put-metric-data commands to add data for the first hour. Replace the example time stamp with a time stamp that is two hours in the past, in Universal Coordinated Time (UTC).

   ```
   aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
   --timestamp 2016-10-14T20:30:00Z --value 87 --unit Milliseconds
   aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
   --timestamp 2016-10-14T20:30:00Z --value 51 --unit Milliseconds
   aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
   --timestamp 2016-10-14T20:30:00Z --value 125 --unit Milliseconds
   aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
   --timestamp 2016-10-14T20:30:00Z --value 235 --unit Milliseconds
   ```

2. Add data for the second hour, using a time stamp that is one hour later than the first hour.

   ```
   aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
   --timestamp 2016-10-14T21:30:00Z --statistic-values
    Sum=577,Minimum=65,Maximum=189,SampleCount=5 --unit Milliseconds
   ```

3. Add data for the third hour, omitting the time stamp to default to the current time.

   ```
   aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted \
   --statistic-values Sum=806,Minimum=47,Maximum=328,SampleCount=6 --unit Milliseconds
   ```

Step 3: Get Statistics from CloudWatch

Now that you have published metrics to CloudWatch, you can retrieve statistics based on those metrics using the get-metric-statistics command as follows. Be sure to specify `--start-time` and `--end-time` far enough in the past to cover the earliest time stamp that you published.

```
aws cloudwatch get-metric-statistics --namespace GetStarted --metric-name RequestLatency --
statistics Average \
--start-time 2016-10-14T00:00:00Z --end-time 2016-10-15T00:00:00Z --period 60
```

The following is example output:

```
{
 "Datapoints": [],
 "Label": "Request:Latency"
}
```

Step 4: View Graphs with the Console

After you have published metrics to CloudWatch, you can use the CloudWatch console to view statistical graphs.

To view graphs of your statistics on the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, choose **Metrics**.
3. In the **All metrics** tab, in the search box, type **RequestLatency** and press Enter.
4. Select the check box for the **RequestLatency** metric. A graph of the metric data is displayed in the upper pane.

For more information, see .

Using Amazon CloudWatch Dashboards

Amazon CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources.

With dashboards, you can create the following:

- A single view for selected metrics and alarms to help you assess the health of your resources and applications across one or more regions. You can select the color used for each metric on each graph, so that you can easily track the same metric across multiple graphs.
- An operational playbook that provides guidance for team members during operational events about how to respond to specific incidents.
- A common view of critical resource and application measurements that can be shared by team members for faster communication flow during operational events.

You can create dashboards by using the console, the AWS CLI, or by using the `PutDashboard` API.

Contents

Create a CloudWatch Dashboard

To get started with CloudWatch dashboards, you must first create a dashboard. You can create multiple dashboards. There is no limit on the number of CloudWatch dashboards in your AWS account. All dashboards are global, not region-specific.

The steps in this section are for creating a dashboard using the console. You can also create a dashboard with the `PutDashboard` API, which uses a JSON string to define the dashboard contents. To create a dashboard using `PutDashboard` and base this dashboard on an existing dashboard, choose **Actions**, **View/edit source** to display and copy the JSON string of a current dashboard to use for your new dashboard.

For more information about creating a dashboard using the API, see PutDashboard in the Amazon CloudWatch API Reference.

To create a dashboard using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards**, **Create dashboard**.

3. In the **Create new dashboard** dialog box, type a name for the dashboard and choose **Create dashboard**.

4. Do one of the following in the **Add to this dashboard** dialog box:

    *   To add a graph to your dashboard, choose **Line** or **Stacked area** and choose **Configure**. In the **Add metric graph** dialog box, select the metrics to graph and choose **Create widget**. If a specific metric does not appear in the dialog box because it has not published data in more than 14 days, you can add it manually. For more information, see Graph Metrics Manually on a CloudWatch Dashboard (p. 24).

    *   To add a number displaying a metric to the dashboard, choose **Number**, **Configure**. In the **Add metric graph** dialog box, select the metrics to graph and choose **Create widget**.

    *   To add a text block to your dashboard, choose **Text**, **Configure**. In the **New text widget** dialog box, for **Markdown**, add and format your text using Markdown. Choose **Create widget**.

5. Optionally, choose **Add widget** and repeat step 4 to add another widget to the dashboard. You can repeat this step multiple times.

6. Choose **Save dashboard**.

Add or Remove a Graph from a CloudWatch Dashboard

You can add graphs containing one or more metrics to your dashboard for the resources you monitor. You can remove the graphs when they're no longer needed.

To add a graph to a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards** and select a dashboard.

3. Choose **Add widget**.

4. Choose either **Line** or **Stacked area** and choose **Configure**.

5. In the **All metrics** tab, select the metrics to graph. If a specific metric does not appear in the dialog box because it has not published data in more than 14 days, you can add it manually. For more information, see Graph Metrics Manually on a CloudWatch Dashboard (p. 24).

6. (Optional) As you choose metrics to graph, you can change their color on the graph. To do so, choose **Graphed metrics** and select the color square next to the metric to display a color picker box. Choose another color square in the color picker. Click outside the color picker to see your new color on the graph. Alternatively, in the color picker, you can type the six-digit standard HTML hex color code for the color you want and press ENTER.

7. (Optional) To view more information about the metric being graphed, hover over the legend.

8. (Optional) To change the widget type, hover over the title area of the graph and choose **Widget actions**, **Widget type**.

9. (Optional) To change the statistic used for a metric, choose **Graphed metrics**, **Statistic**, and select the statistic you want to use. For more information, see Statistics (p. 5).

10. (Optional) To change the time range shown on the graph, choose either **custom** at the top of the graph, or one of the time periods to the left of **custom**.

11. (Optional) Horizontal annotations can help dashboard users quickly see when a metric has spiked to a certain level, or whether the metric is within a predefined range. To add a horizontal annotation, choose **Graph options**, **Add horizontal annotation**:

    a. For **Label**, type a label for the annotation.

    b. For **Value**, type the metric value where the horizontal annotation appears.

    c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose `Above` or `Below` for the corresponding area to be filled. If you specify `Between`, another `Value` field appears, and the area of the graph between the two values is filled.

    d. For **Axis**, specify whether the numbers in `Value` refer to the metric associated with the left Y-axis or the right Y-axis, if the graph includes multiple metrics.

       You can change the fill color of an annotation by choosing the color square in the left column of the annotation.

    Repeat these steps to add multiple horizontal annotations to the same graph.

    To hide an annotation, clear the check box in the left column for that annotation.

    To delete an annotation, choose **x** in the **Actions** column.

12. (Optional) Vertical annotations help you mark milestones in a graph, such as operational events or the beginning and end of a deployment. To add a vertical annotation, choose **Graph options**, **Add vertical annotation**:

    a. For **Label**, type a label for the annotation. To show only the date and time on the annotation, keep the **Label** field blank.

    b. For **Date**, specify the date and time where the vertical annotation appears.

    c. For **Fill**, specify whether to use fill shading before or after a vertical annotation, or between two vertical annotations. For example, choose `Before` or `After` for the corresponding area to be filled. If you specify `Between`, another `Date` field appears, and the area of the graph between the two values is filled.

    Repeat these steps to add multiple vertical annotations to the same graph.

    To hide an annotation, clear the check box in the left column for that annotation.

    To delete an annotation, choose **x** in the **Actions** column.

13. Choose **Create widget**.

14. Choose **Save dashboard**.

To remove a graph from a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards** and select a dashboard.

3. Hover over the title of the graph and choose **Widget actions**, **Delete**.

4.   Choose **Save dashboard**. If you attempt to navigate away from the dashboard before you save your changes, you are prompted to either save or discard your changes.

Move or Resize a Graph on a CloudWatch Dashboard

You can arrange and resize graphs on your CloudWatch dashboard.

To move a graph on a dashboard

1.   Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2.   In the navigation pane, choose **Dashboards** and select a dashboard.
3.   Hover over the title of the graph until the selection icon appears. Select and drag the graph to a new location on the dashboard.
4.   Choose **Save dashboard**.

To resize a graph

1.   Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2.   In the navigation pane, choose **Dashboards** and select a dashboard.
3.   To increase or decrease the size, hover over the graph and drag the lower right corner of the graph.
4.   Choose **Save dashboard**.

To enlarge a graph temporarily

1.   Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2.   In the navigation pane, choose **Dashboards** and select a dashboard.
3.   Select the graph. Alternatively, hover over the title of the graph and choose **Widget actions**, **Enlarge**.

Edit a Graph on a CloudWatch Dashboard

You can edit a graph to change the title, statistic, or period, or to add or remove metrics. If you have multiple metrics displayed on a graph, you can reduce the clutter by temporarily hiding the metrics that don't interest you.

To edit a graph on a dashboard

1.   Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2.   In the navigation pane, choose **Dashboards** and select a dashboard.
3.   Hover over the title of the graph and choose **Widget actions**, **Edit**.
4.   To change the graph's title, select the title, type a new title, and press ENTER.
5.   To change the time range shown on the graph, choose either **custom** at the top of the graph, or one of the time periods to the left of **custom**.
6.   To change the type of widget between separate lines on a graph, stacked lines on a graph, and a number, choose the box next to the right of **custom** and select either **Line**, **Stacked area**, or **Number**.

7. In the lower half of the screen, in the **Graphed metrics** tab, you can change the colors, statistic, or period:

    a. To change the color of one of the lines, select the color square next to the metric to display a color picker box. Choose another color in the color picker, and click outside the color picker to see your new color on the graph. Alternatively, in the color picker, you can type the six-digit HTML hex color code for the color you want and press ENTER.

    b. To change the statistic, choose **Statistic** in the lower half of the window, and choose the new statistic you want. For more information, see Statistics (p. 5).

    c. To change the time period, which is next to **Statistic** in the lower half of the window, choose **Period** and select another value. This new setting is used on the dashboard only if the period setting of the dashboard itself is set to `Auto`. Otherwise, the period setting of the dashboard overrides the period setting for individual widgets.

8. To add or edit horizontal annotations, choose **Graph options**:

    a. To add a horizontal annotation, choose **Add horizontal annotation**.

    b. For **Label**, type a label for the annotation.

    c. For **Value**, type the metric value where the horizontal annotation appears.

    d. For **Fill**, specify how to use fill shading with this annotation. For example, choose `Above` or `Below` for the corresponding area to be filled. If you specify `Between`, another `Value` field appears, and the area of the graph between the two values is filled.

        You can change the fill color of an annotation by choosing the color square in the left column of the annotation.

    e. For **Axis**, specify whether the numbers in `Value` refer to the metric associated with the left Y-axis or the right Y-axis, if the graph includes multiple metrics.

    Repeat these steps to add multiple horizontal annotations to the same graph.

    To hide an annotation, clear the check box in the left column for that annotation.

    To delete an annotation, click the **x** in the **Actions** column.

9. To add or edit vertical annotations, choose **Graph options**: **Add vertical annotation**:

    a. To add a vertical annotation, choose **Add vertical annotation**.

    b. For **Label**, type a label for the annotation. To show only the date and time on the annotation, keep the **Label** field blank.

    c. For **Date**, specify the date and time where the vertical annotation appears.

    d. For **Fill**, specify whether to use fill shading before or after a vertical annotation, or between two vertical annotations. For example, choose `Before` or `After` for the corresponding area to be filled. If you specify `Between`, another `Date` field appears, and the area of the graph between the two values is filled.

    Repeat these steps to add multiple vertical annotations to the same graph.

    To hide an annotation, clear the check box in the left column for that annotation.

    To delete an annotation, choose **x** in the **Actions** column.

10. To hide or change the position of the graph legend, hover over the title of the graph and choose **Widget actions**, **Edit**. Hover over **Legend** and choose **Hidden**, **Bottom**, or **Right**.

11. To customize the Y-axis, choose **Graph options**. You can type a custom label in **Label** under **Left Y Axis**. If the graph also displays values on the right Y-axis, you can customize that label as well. You can also set minimums and maximums on the Y-axis values, and the graph displays only the value range you specify.

12. When you're finished with your changes, choose **Update widget**.

To temporarily hide metrics for a graph on a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. In the graph's footer, hover over the colored square in the legend. When it changes to an X, click it.
4. To restore the metric, choose the grayed out square and metric name.

Graph Metrics Manually on a CloudWatch Dashboard

If a metric has not published data in the past 14 days, you cannot find it when searching for metrics to add to a graph on a CloudWatch dashboard. Use the following steps to add any metric manually to an existing graph.

To add a metric that you cannot find in search to a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. The dashboard must already contain a graph where you want to add the metric. If it does not already, create the graph and add any metric to it. For more information, see Add or Remove a Graph from a CloudWatch Dashboard (p. 20).
4. Choose **Actions**, **View/edit source**.

   A JSON block appears. The block specifies the widgets on the dashboard and their contents. The following is an example of one part of this block, which defines one graph.

   ```
   {
               "type": "metric",
               "x": 0,
               "y": 0,
               "width": 6,
               "height": 3,
               "properties": {
                   "view": "singleValue",
                   "metrics": [
                       [ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
                   ],
                   "region": "us-west-1"
               }
           },
   ```

   In this example, the following section defines the metric shown on this graph.

   ```
   [ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
   ```

5. Add a comma after the end bracket if there is not already one, and then add a similar bracketed section after the comma. In this new section, specify the namespace, metric name, and any necessary dimensions of the metric you are adding to the graph. The following is an example:

   ```
   [ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ],
   [ "MyNamespace", "MyMetricName", "DimensionName", "DimensionValue" ]
   ```

For more information about the formatting of metrics in JSON, see Properties of a Metric Widget Object.

6. Choose **Update**.

Rename a Graph on a CloudWatch Dashboard

You can change the default name that CloudWatch assigns to a graph on your dashboard.

To rename a graph on a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Hover over the title of the graph and choose **Widget actions**, **Edit**.
4. On the **Edit graph** screen, near the top, choose the title of the graph.
5. For **Title**, type a new name and choose **Ok** (check mark). In the lower-right corner of the **Edit graph** screen, choose **Update widget**.

Add or Remove a Text Widget from a CloudWatch Dashboard

A text widget contains a block of text in Markdown format. You can add, edit, or remove text widgets from your CloudWatch dashboard.

To add a text widget to a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Choose **Add widget**.
4. Choose **Text**, **Configure**.
5. For **Markdown**, add and format your text using Markdown and choose **Create widget**.
6. Choose **Save dashboard**.

To edit a text widget on a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Hover over the upper-right corner of the text block and choose **Widget actions**, **Edit**.
4. Update the text as needed and choose **Update widget**.
5. Choose **Save dashboard**.

To remove a text widget from a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Hover over the upper-right corner of the text block and choose **Widget actions**, **Delete**.

4. Choose **Save dashboard**.

Add or Remove an Alarm from a CloudWatch Dashboard

You can add alarms that you have created to your dashboard. When an alarm is on a dashboard, it turns red when it is in the `ALARM` state.

To add an alarm to a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, select the alarm to add, and then choose **Add to Dashboard**.
3. Select a dashboard, choose a widget type (**Line**, **Stacked area**, or **Number**), and then choose **Add to dashboard**.
4. To see your alarm on the dashboard, choose **Dashboards** in the navigation pane and select the dashboard.
5. (Optional) To temporarily make an alarm graph larger, select the graph.
6. (Optional) To change the widget type, hover over the title of the graph and choose **Widget actions**, **Widget type**.

To remove an alarm from a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Hover over the title of the graph and choose **Widget actions**, **Delete**.
4. Choose **Save dashboard**. If you attempt to navigate away from the dashboard before you save your changes, you are prompted to either save or discard your changes.

Monitor Resources in Multiple Regions Using a CloudWatch Dashboard

You can monitor AWS resources in multiple Regions using a single CloudWatch dashboard. For example, you can create a dashboard that shows CPU utilization for an EC2 instance located in the `us-west-2` Region with your billing metrics, which are located in the `us-east-1` Region.

To monitor resources in multiple Regions in one dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. In the navigation bar, select a Region.
4. Select the metrics to add to your dashboard.
5. For **Actions**, choose **Add to dashboard**.
6. For **Add to**, type a name for the new dashboard and choose **Add to dashboard**.

   Alternatively, to add to an existing dashboard, choose **Existing dashboard**, select a dashboard, and then choose **Add to dashboard**.

7. To add metrics from another Region, select the next Region and repeat these steps.

8. Choose **Save dashboard**.

Link and Unlink Graphs on a CloudWatch Dashboard

You can link the graphs on your dashboard together, so that when you zoom in or zoom out on one graph, the other graphs zoom in or zoom out at the same time. You can unlink graphs to limit zoom to one graph.

To link the graphs on a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards** and select a dashboard.

3. Choose **Actions**, **Link graphs**.

To unlink the graphs on a dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards** and select a dashboard.

3. Clear **Actions**, **Link graphs**.

Add a Dashboard to Your Favorites List

You can add a CloudWatch dashboard to a list of favorite dashboards, to help you find it quickly. The **Favorites** list appears at the bottom of the navigation pane.

To add a dashboard to the Favorites list

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards**.

3. Select the star symbol next to the dashboard to add.

Change the Refresh Interval for the CloudWatch Dashboard

You can change how often the data on your CloudWatch dashboard is refreshed or set it to automatically refresh.

To change the dashboard refresh interval

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Dashboards** and select a dashboard.

3. On the **Refresh options** menu (upper right corner), choose **10 Seconds**, **1 Minute**, **2 Minutes**, **5 Minutes**, or **15 Minutes**.

To automatically refresh the dashboard

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Choose **Refresh options**, **Auto refresh**.

Change the Time Range or Time Zone Format of a CloudWatch Dashboard

You can change the time range to display dashboard data over minutes, hours, days, or weeks. You can also change the time format to display dashboard data in UTC or local time.

> **Note**
> If you create a dashboard with graphs that contain close to 100 or more high-resolution metrics, we recommend that you set the time range to no longer than one hour, to ensure good dashboard performance. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

To change the dashboard time range

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Do one of the following:
    - Select one of the predefined ranges shown, which span from 1 hour to 1 week: 1h, 3h, 12h, 1d, 3d, or 1w.
    - Choose **custom**, **Relative**. Select one of the predefined ranges, which span from 1 minute to 15 months.
    - Choose **custom**, **Absolute**. Use the calendar picker or the text fields to specify the time range.

    > **Note**
    > When you change the time range of a graph while the aggregation period is set to **Auto**, CloudWatch may change the period. To manually set the period, choose **Actions** and select a new value for **Period:**.

To change the dashboard time format

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Choose **custom**.
4. From the upper corner, choose **UTC** or **Local timezone**.

Using Amazon CloudWatch Metrics

Metrics are data about the performance of your systems. By default, several services provide free metrics for resources (such as Amazon EC2 instances, Amazon EBS volumes, and Amazon RDS DB instances). You can also enable detailed monitoring some resources, such as your Amazon EC2 instances, or publish your own application metrics. Amazon CloudWatch can load all the metrics in your account (both AWS resource metrics and application metrics that you provide) for search, graphing, and alarms.

Metric data is kept for 15 months, enabling you to view both up-to-the-minute data and historical data.

Contents

View Available Metrics

Metrics are grouped first by namespace, and then by the various dimension combinations within each namespace. For example, you can view all EC2 metrics, EC2 metrics grouped by instance, or EC2 metrics grouped by Auto Scaling group.

Only the AWS services that you're using send metrics to Amazon CloudWatch.

To view available metrics by namespace and dimension using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select a metric namespace (for example, EC2).
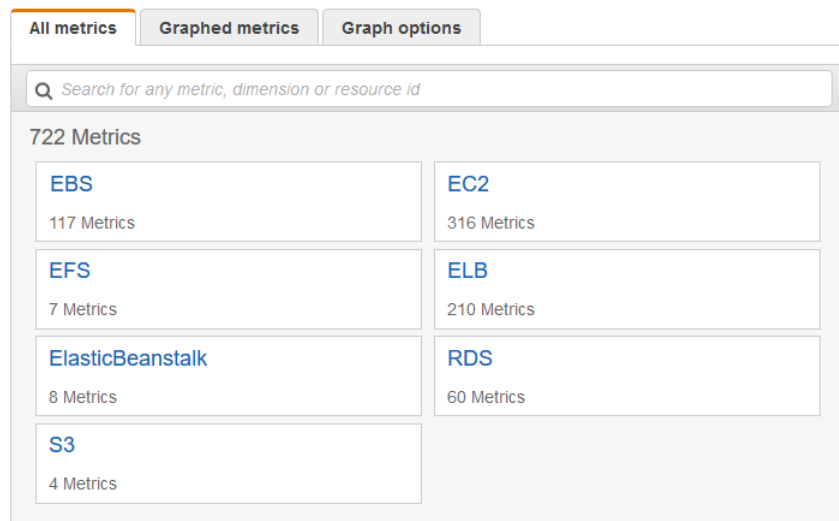
4. Select a metric dimension (for example, Per-Instance Metrics).
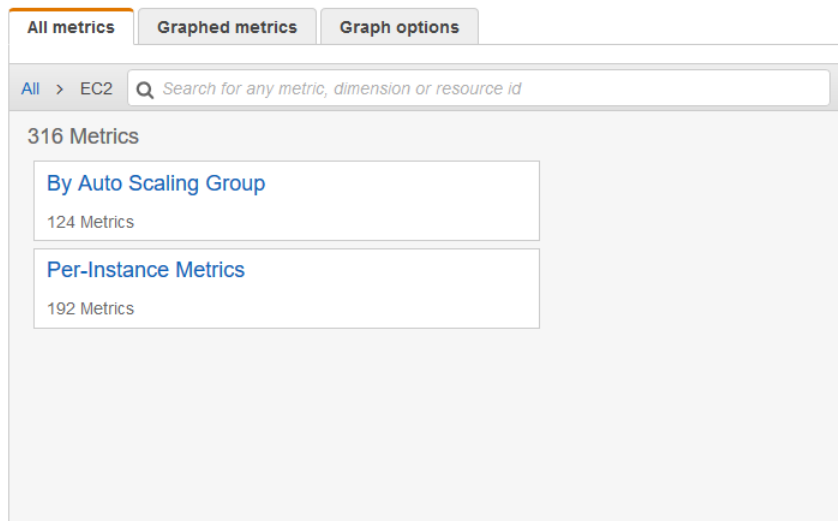
5. The **All metrics** tab displays all metrics for that dimension in the namespace. You can do the following:

   a. To sort the table, use the column heading.

   b. To graph a metric, select the check box next to the metric. To select all metrics, select the check box in the heading row of the table.

   c. To filter by resource, choose the resource ID and then choose **Add to search**.

   d. To filter by metric, choose the metric name and then choose **Add to search**.
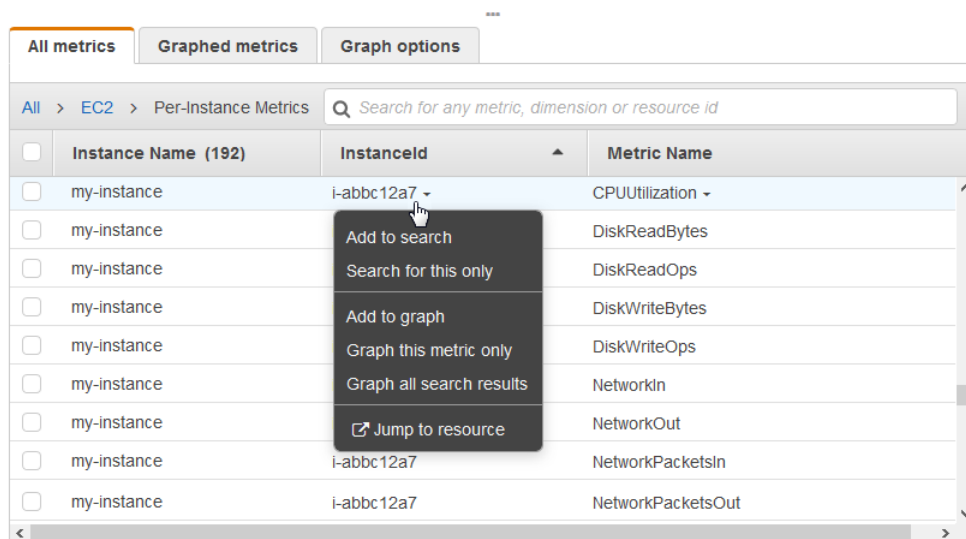
To view available metrics by namespace, dimension, or metric using the AWS CLI

Use the list-metrics command to list CloudWatch metrics. For a list of all service namespaces, see AWS Namespaces (p. 115). For lists of the metrics and dimensions for each service, see Amazon CloudWatch Metrics and Dimensions Reference (p. 114).

The following example specifies the `AWS/EC2` namespace to view all the metrics for Amazon EC2:

```
aws cloudwatch list-metrics --namespace AWS/EC2
```

The following is example output:

```
{
  "Metrics" : [
    ...
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkOut"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "CPUUtilization"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkIn"
    },
    ...
  ]
}
```

To list all the available metrics for a specified resource

The following example specifies the `AWS/EC2` namespace and the `InstanceId` dimension to view the results for the specified instance only.

```
aws cloudwatch list-metrics --namespace AWS/EC2 --dimensions
 Name=InstanceId,Value=i-1234567890abcdef0
```

To list a metric for all resources

The following example specifies the `AWS/EC2` namespace and a metric name to view the results for the specified metric only.

```
aws cloudwatch list-metrics --namespace AWS/EC2 --metric-name CPUUtilization
```

Search for Available Metrics

You can search within all the metrics in your account using targeted search terms. Metrics are returned that have matching results within their namespace, metric name, or dimensions.

To search for available metrics in CloudWatch

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. In the search field on the **All metrics** tab, type a search term, such as a metric name, service name, or resource name, and press Enter. This shows you all the namespaces with metrics with this search term.

    For example, if you search for `volume`, this shows the namespaces that contain metrics with this term in their name.
4. Select a namespace with results for your search to view the metrics. You can do the following:

    a. To graph one or more metrics, select the check box next to each metric. To select all metrics, select the check box in the heading row of the table.
    b. To view one of the resources in its console, choose the resource ID and then choose **Jump to resource**.
    c. To view help for a metric, select the metric name and choose **What is this?**.
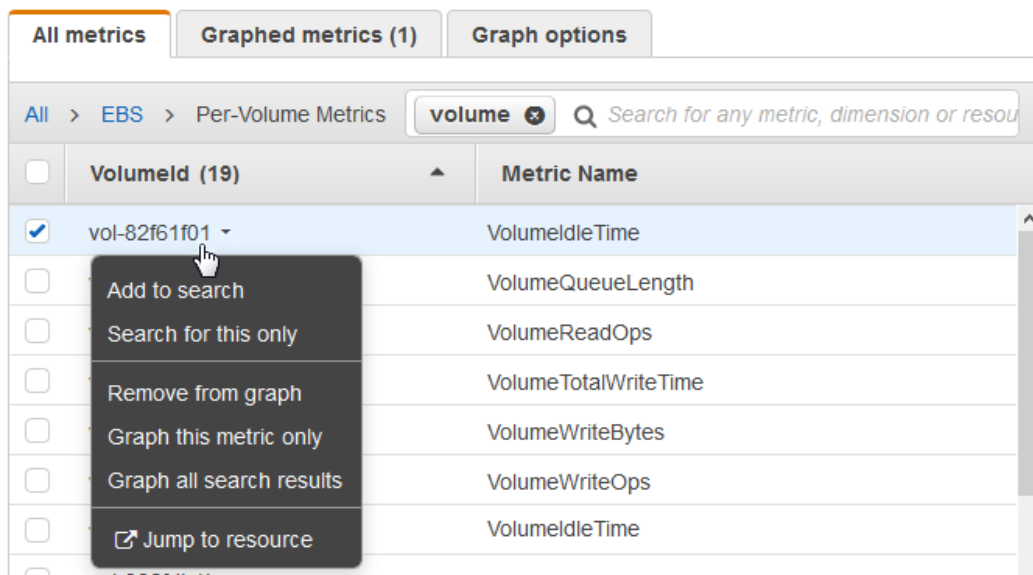
Get Statistics for a Metric

The following examples show you how to get statistics for the CloudWatch metrics for your resources, such as your EC2 instances.

Examples

Get Statistics for a Specific Resource

The following example shows you how to determine the maximum CPU utilization of a specific EC2 instance.

Requirements

- You must have the ID of the instance. You can get the instance ID using the Amazon EC2 console or the describe-instances command.
- By default, basic monitoring is enabled, but you can enable detailed monitoring. For more information, see Enable or Disable Detailed Monitoring for Your Instances in the *Amazon EC2 User Guide for Linux Instances*.

To display the average CPU utilization for a specific instance using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the EC2 metric namespace.
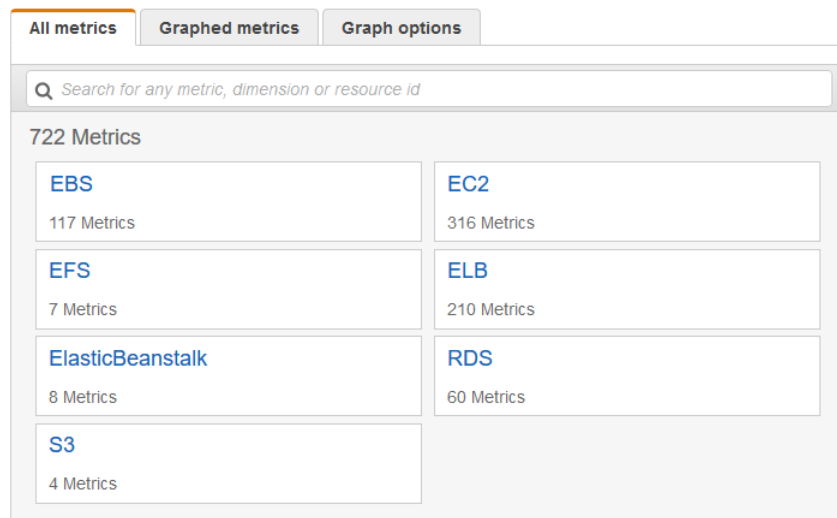
4. Select the Per-Instance Metrics dimension.
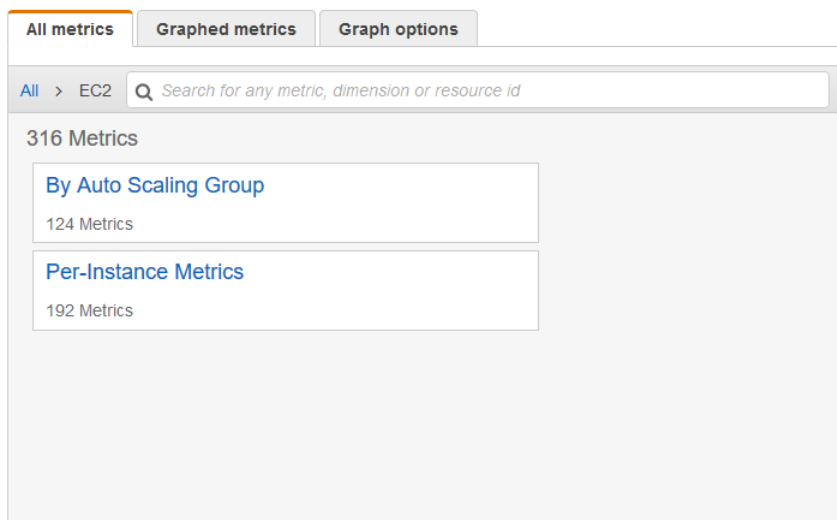
5. In the search field, type **CPUUtilization** and press Enter. Select the row for the specific instance, which displays a graph for the **CPUUtilization** metric for the instance. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.
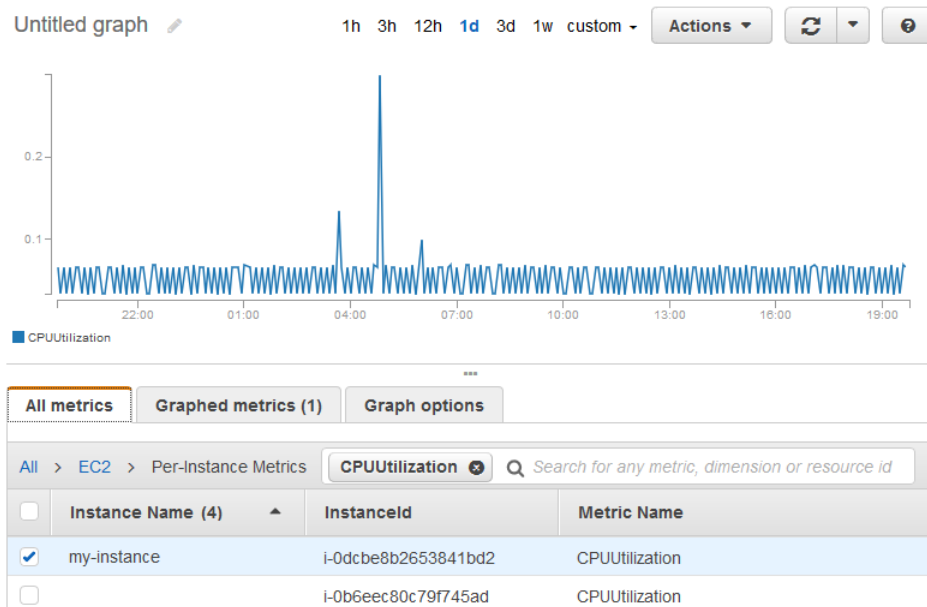
6. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).
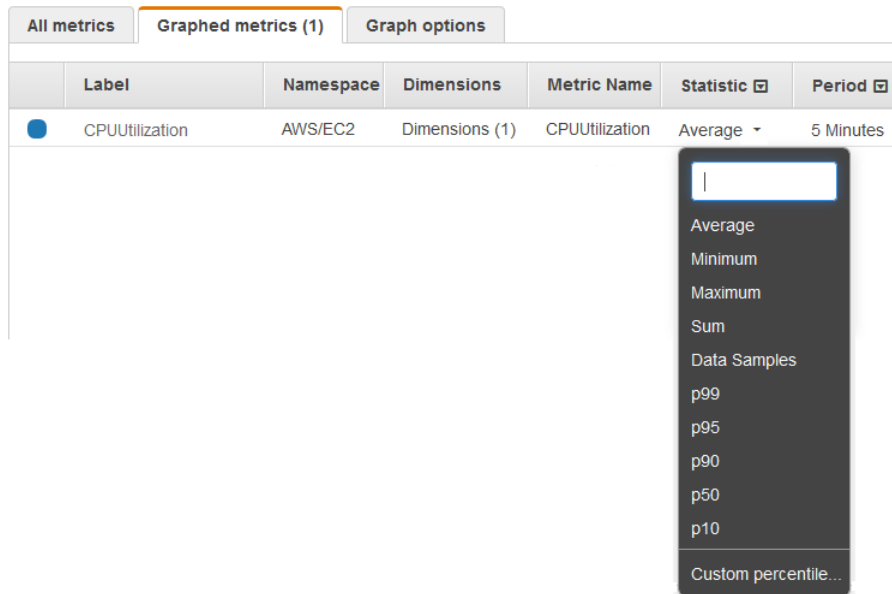
7. To change the period, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose a different value.

To get the CPU utilization per EC2 instance using the AWS CLI

Use the get-metric-statistics command as follows to get the **CPUUtilization** metric for the specified instance:

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-1234567890abcdef0 --statistics Maximum \
--start-time 2016-10-18T23:18:00 --end-time 2016-10-19T23:18:00 --period 360
```

The returned statistics are six-minute values for the requested 24-hour time interval. Each value represents the maximum CPU utilization percentage for the specified instance for a particular six-minute time period. The data points are not returned in chronological order. The following shows the beginning of the example output (the full output includes data points for every 6 minutes of the 24-hour period):

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:18:00Z",
            "Maximum": 0.33000000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T03:18:00Z",
            "Maximum": 99.670000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T07:18:00Z",
            "Maximum": 0.34000000000000002,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
```

```
}
```

Aggregate Statistics Across Resources

You can aggregate the metrics for AWS resources across multiple resources. Amazon CloudWatch cannot aggregate data across Regions. Metrics are completely separate between Regions.

For example, you can aggregate statistics for your EC2 instances that have detailed monitoring enabled. Instances that use basic monitoring are not included. Therefore, you must enable detailed monitoring (at an additional charge), which provides data in 1-minute periods. For more information, see Enable or Disable Detailed Monitoring for Your Instances in the *Amazon EC2 User Guide for Linux Instances*.

This example shows you how to get the average CPU usage for your EC2 instances. Because no dimension is specified, CloudWatch returns statistics for all dimensions in the `AWS/EC2` namespace. To get statistics for other metrics, see Amazon CloudWatch Metrics and Dimensions Reference (p. 114).

> **Important**
> This technique for retrieving all dimensions across an AWS namespace does not work for custom namespaces that you publish to Amazon CloudWatch. With custom namespaces, you must specify the complete set of dimensions that are associated with any given data point to retrieve statistics that include the data point.

To display average CPU utilization for your EC2 instances

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Choose the **EC2** namespace and choose **Across All Instances**.
4. Select the row that contains **CPUUtilization**, which displays a graph for the metric for all your EC2 instances. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.
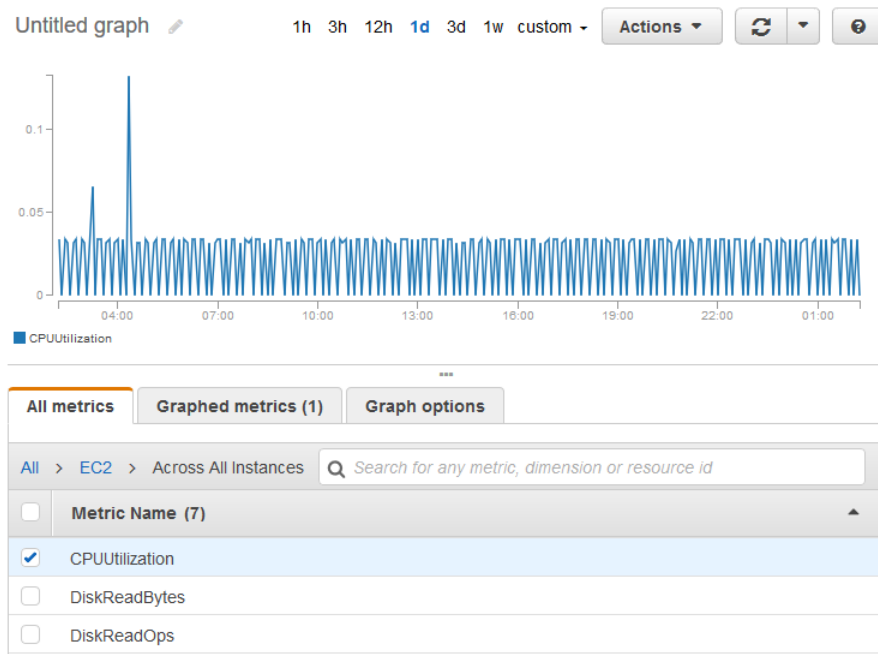
5. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).

6. To change the period, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose a different value.

To get average CPU utilization across your EC2 instances using the AWS CLI

Use the get-metric-statistics command as follows:

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization --
statistics "Average" "SampleCount" \
--start-time 2016-10-11T23:18:00 --end-time 2016-10-12T23:18:00 --period 3600
```

The following is example output:

```
{
    "Datapoints": [
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-12T07:18:00Z",
            "Average": 0.038235294117647062,
            "Unit": "Percent"
        },
        {
            "SampleCount": 240.0,
            "Timestamp": "2016-10-12T09:18:00Z",
            "Average": 0.16670833333333332,
            "Unit": "Percent"
        },
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-11T23:18:00Z",
            "Average": 0.041596638655462197,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

Aggregate Statistics by Auto Scaling Group

You can aggregate statistics for the EC2 instances in an Auto Scaling group. Amazon CloudWatch cannot aggregate data across Regions. Metrics are completely separate between Regions.

This example shows you how to get the total bytes written to disk for one Auto Scaling group. The total is computed for one-minute periods for a 24-hour interval across all EC2 instances in the specified Auto Scaling group.

To display DiskWriteBytes for the instances in an Auto Scaling group using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Metrics**.

3. Choose the **EC2** namespace and then choose **By Auto Scaling Group**.

4. Select the row for the **DiskWriteBytes** metric and the specific Auto Scaling group, which displays a graph for the metric for the instances in the Auto Scaling group. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.
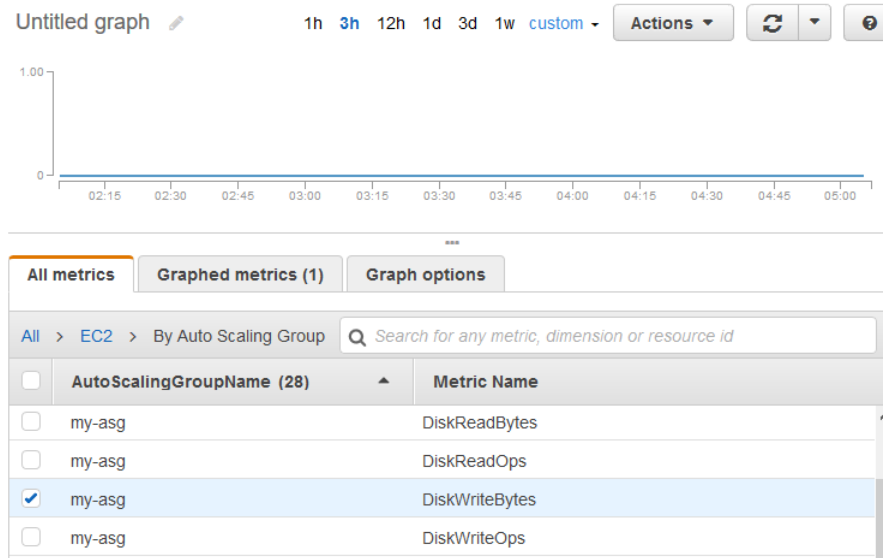
5. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).

6. To change the period, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose a different value.

To get DiskWriteBytes for the instances in an Auto Scaling group using the AWS CLI

Use the get-metric-statistics command as follows:

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name DiskWriteBytes
--dimensions Name=AutoScalingGroupName,Value=my-asg --statistics "Sum" "SampleCount" \
--start-time 2016-10-16T23:18:00 --end-time 2016-10-18T23:18:00 --period 360
```

The following is example output:

```
{
    "Datapoints": [
        {
            "SampleCount": 18.0,
            "Timestamp": "2016-10-19T21:36:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        },
        {
            "SampleCount": 5.0,
            "Timestamp": "2016-10-19T21:42:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "DiskWriteBytes"
}
```

Aggregate Statistics by Amazon Machine Image (AMI)

You can aggregate statistics for the EC2 instances that have detailed monitoring enabled. Instances that use basic monitoring are not included. For more information, see Enable or Disable Detailed Monitoring for Your Instances in the *Amazon EC2 User Guide for Linux Instances*.

This example shows you how to determine average CPU utilization for all instances that use the specified AMI. The average is over 60-second time intervals for a one-day period.

To display the average CPU utilization by AMI using the console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Choose the **EC2** namespace and then choose **By Image (AMI) Id**.
4. Select the row for the **CPUUtilization** metric and the specific AMI, which displays a graph for the metric for the specified AMI. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.
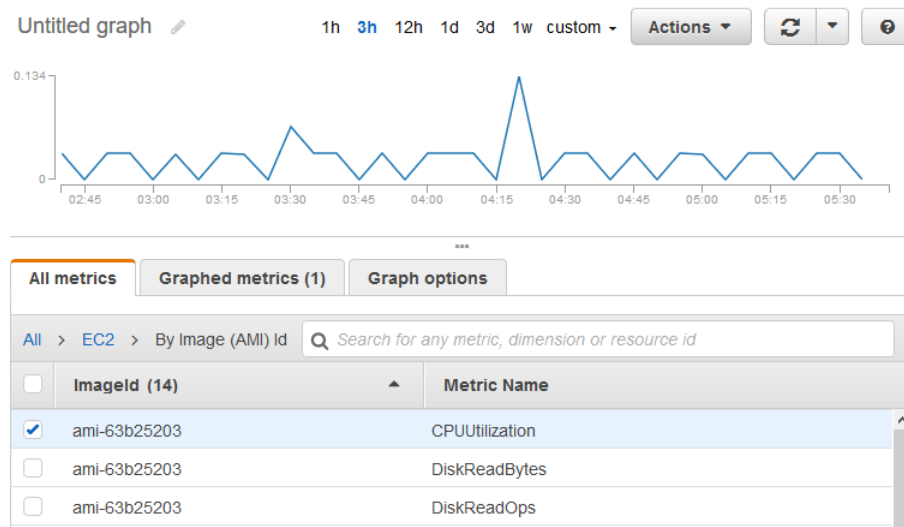
5. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).
6. To change the period, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose a different value.

To get the average CPU utilization by AMI using the AWS CLI

Use the get-metric-statistics command as follows:

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=ImageId,Value=ami-3c47a355 --statistics Average \
--start-time 2016-10-10T00:00:00 --end-time 2016-10-11T00:00:00 --period 3600
```

The operation returns statistics that are one-minute values for the one-day interval. Each value represents an average CPU utilization percentage for EC2 instances running the specified AMI. The following is example output:

```
{
```

```
    "Datapoints": [
        {
            "Timestamp": "2016-10-10T07:00:00Z",
            "Average": 0.041000000000000009,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T14:00:00Z",
            "Average": 0.079579831932773085,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T06:00:00Z",
            "Average": 0.036000000000000011,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

Graph Metrics

You can use the CloudWatch console to graph metric data generated by other AWS services to make it easier to see the metric activity on your services. You can use the following procedures to graph metrics in CloudWatch.

Contents

- Graph a Metric (p. 40)
- Modify the Time Range or Time Zone Format for a Graph (p. 42)
- Modify the Y Axis for a Graph (p. 43)
- Create an Alarm from a Metric on a Graph (p. 44)

Graph a Metric

You can select metrics and create graphs of the data using the CloudWatch console.

CloudWatch supports the following statistics on metrics: `Average`, `Minimum`, `Maximum`, `Sum`, and `SampleCount`. For more information, see Statistics (p. 5).

You can view your data at different granularities. For example, you can choose a detailed view (for example 1 minute), which can be useful when troubleshooting. You can choose a less detailed view (for example, 1 hour), which can be useful when viewing a broader time range (for example, 3 days) so that you can see trends over time. For more information, see Periods (p. 6).

Create a Graph

To graph a metric

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, type a search term in the search field, such as a metric name or resource name, and press Enter.

For example, if you search for the **CPUUtilization** metric, you see the namespaces and dimensions with this metric.

4. Select one of the results for your search to view the metrics.

5. To graph one or more metrics, select the check box next to each metric. To select all metrics, select the check box in the heading row of the table.

6. To view more information about the metric being graphed, hover over the legend.

7. Horizontal annotations can help graph users quickly see when a metric has spiked to a certain level, or whether the metric is within a predefined range. To add a horizontal annotation, choose **Graph options**, **Add horizontal annotation**:

   a. For **Label**, type a label for the annotation.

   b. For **Value**, type the metric value where the horizontal annotation appears.

   c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose `Above` or `Below` for the corresponding area to be filled. If you specify `Between`, another `Value` field appears, and the area of the graph between the two values is filled.

   d. For **Axis**, specify whether the numbers in `Value` refer to the metric associated with the left Y-axis or the right Y-axis, if the graph includes multiple metrics.

      You can change the fill color of an annotation by choosing the color square in the left column of the annotation.

   Repeat these steps to add multiple horizontal annotations to the same graph.

   To hide an annotation, clear the check box in the left column for that annotation.

   To delete an annotation, choose **x** in the **Actions** column.

8. To get a URL for your graph, choose **Actions**, **Share**. Copy the URL and save it or share it.

9. To add your graph to a dashboard, choose **Actions**, **Add to dashboard**.

Update a Graph

To update your graph

1. To change the name of the graph, choose the pencil icon.

2. To change the time range, select one of the predefined values or choose **custom**. For more information, see .

3. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).

4. To change the period, choose the **Graphed metrics** tab. Choose the column heading or an individual value, and then choose a different value.

5. To add a horizontal annotation, choose **Graph options**, **Add horizontal annotation**:

   a. For **Label**, type a label for the annotation.

   b. For **Value**, type the metric value where the horizontal annotation appears.

   c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose `Above` or `Below` for the corresponding area to be filled. If you specify `Between`, another `Value` field appears, and the area of the graph between the two values is filled.

   d. For **Axis**, specify whether the numbers in `Value` refer to the metric associated with the left Y-axis or the right Y-axis, if the graph includes multiple metrics.

You can change the fill color of an annotation by choosing the color square in the left column of the annotation.

Repeat these steps to add multiple horizontal annotations to the same graph.

To hide an annotation, clear the check box in the left column for that annotation.

To delete an annotation, choose **x** in the **Actions** column.

6. To change the refresh interval, choose **Refresh options**, and then select **Auto refresh** or choose **1 Minute**, **2 Minutes**, **5 Minutes**, or **15 Minutes**.

Duplicate a Metric

To duplicate a metric

1. Choose the **Graphed metrics** tab.
2. For **Actions**, choose the **Duplicate** icon.
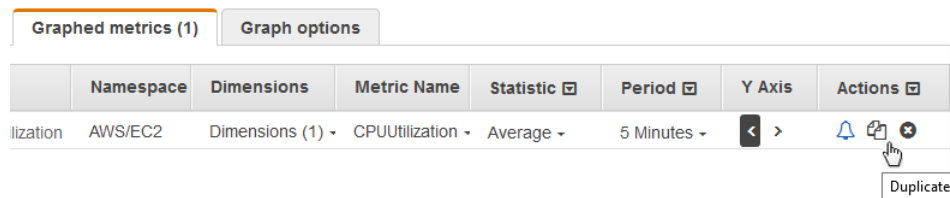
3. Update the duplicate metric as needed.

Modify the Time Range or Time Zone Format for a Graph

You can change the time range or the time zone format of a graph.

Relative Time Ranges

You can set a relative time range for your graph.

To specify a relative time range for a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select one of the predefined ranges shown at the top of the page, which span from 1 hour to 1 week ago.
4. For more predefined ranges, choose the **custom** menu and then choose **Relative**. Select one of the predefined ranges, which span from 5 minutes to 15 months ago.

Absolute Time Ranges

You can set an absolute time range for your graph.

To specify an absolute time range for a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Metrics**.

3. Choose the **custom** menu and then choose **Absolute**. Use the calendar picker or the text fields to specify the time range.

Setting the Time Zone Format

You can specify whether the graph uses UTC time or your local time.

To specify the time zone for a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Metrics**.

3. Choose the **custom** menu and then choose **UTC** or **Local timezone**.

Zoom in on a Graph

You can change the granularity of a graph and zoom in to see data over a shorter time period.

To zoom in on a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Metrics**.

3. Choose and drag on the graph area, and then release the drag.

4. To reset a zoomed-in graph, choose the **Reset zoom** icon.

Modify the Y Axis for a Graph

You can set custom bounds for the Y axis on a graph to help you see the data better. For example, you can change the bounds on a CPUUtilization graph to 100 percent so that it's easy to see whether the CPU is low (the plotted line is near the bottom of the graph) or high (the plotted line is near the top of the graph).

You can switch between two different Y axes for your graph. This is useful if the graph contains metrics that have different units or that differ greatly in their range of values.

To modify the Y axis on a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Metrics**.

3. Select a metric namespace (for example, EC2) and then a metric dimension (for example, Per-Instance Metrics).

4. The **All metrics** tab displays all metrics for that dimension in that namespace. To graph a metric, select the check box next to the metric.

5. On the **Graph options** tab, specify the **Min** and **Max** values for **Left Y Axis**. The value of **Min** cannot be greater than the value of **Max**.

6. To create a second Y axis, specify the **Min** and **Max** values for **Right Y Axis**.

7. To switch between the two Y axes, choose the **Graphed metrics** tab. For **Y Axis**, choose **Left Y Axis** or **Right Y Axis**.

Create an Alarm from a Metric on a Graph

You can graph a metric and then create an alarm from the metric on the graph, which has the benefit of populating many of the alarm fields for you.
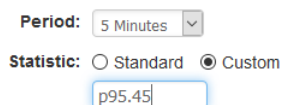
To create an alarm from a metric on a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Metrics**.

3. Select a metric namespace (for example, EC2) and then a metric dimension (for example, Per-Instance Metrics).

4. The **All metrics** tab displays all metrics for that dimension in that namespace. To graph a metric, select the check box next to the metric.

5. To create an alarm for the metric, choose the **Graphed metrics** tab. For **Actions**, choose the alarm icon.

6. Under **Alarm Threshold**, type a unique name for the alarm and a description of the alarm. For **Whenever**, specify a threshold and the number of periods.

7. Under **Actions**, select the type of action to have the alarm perform when the alarm is triggered.

8. (Optional) For **Period**, choose a different value. For **Statistic**, choose **Standard** to specify one of the statistics in the list or choose **Custom** to specify a percentile (for example, p95.45).

9. Choose **Create Alarm**.

Publish Custom Metrics

You can publish your own metrics to CloudWatch using the AWS CLI or an API. You can view statistical graphs of your published metrics with the AWS Management Console.

CloudWatch stores data about a metric as a series of data points. Each data point has an associated time stamp. You can even publish an aggregated set of data points called a *statistic set*.

Topics
- High-Resolution Metrics (p. 45)
- Using Dimensions (p. 45)
- Publish Single Data Points (p. 46)
- Publish Statistic Sets (p. 47)
- Publish the Value Zero (p. 47)

High-Resolution Metrics

Each metric is one of the following:

- Standard resolution, with data having a one-minute granularity
- High resolution, with data at a granularity of one second

Metrics produced by AWS services are standard resolution by default. When you publish a custom metric, you can define it as either standard resolution or high resolution. When you publish a high-resolution metric, CloudWatch stores it with a resolution of 1 second, and you can read and retrieve it with a period of 1 second, 5 seconds, 10 seconds, 30 seconds, or any multiple of 60 seconds.

High-resolution metrics can give you more immediate insight into your application's sub-minute activity. Keep in mind that every `PutMetricData` call for a custom metric is charged, so calling `PutMetricData` more often on a high-resolution metric can lead to higher charges. For more information about CloudWatch pricing, see Amazon CloudWatch Pricing.

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. There is a higher charge for high-resolution alarms with a period of 10 or 30 seconds.

Using Dimensions

In custom metrics, the `--dimensions` parameter is common. A dimension further clarifies what the metric is, and what data it stores. You can have up to 10 dimensions in one metric, and each dimension is defined by a Name and Value pair.

How you specify a dimension is different when you use different commands. With put-metric-data, you specify each dimension as *MyName*=*MyValue*, while with get-metric-statistics or put-metric-alarm you use the format Name=*MyName*, Value=*MyValue*. For example, the following command publishes a "Buffers" metric with two dimensions named InstanceId and InstanceType.

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace MyNameSpace --unit Bytes
 --value 231434333 --dimensions InstanceId=1-23456789,InstanceType=m1.small
```

This command retrieves statistics for that same metric. Separate the Name and Value parts of a single dimension with commas, but you use a space between one dimension and the next if you have multiple dimensions.

```
aws cloudwatch get-metric-statistics --metric-name Buffers --namespace MyNameSpace --
dimensions Name=InstanceId,Value=1-23456789 Name=InstanceType,Value=m1.small --start-time
 2016-10-15T04:00:00Z --end-time 2016-10-19T07:00:00Z --statistics Average --period 60
```

If a single metric includes multiple dimensions, you must specify a value for every defined dimension when you use get-metric-statistics. For example, the Amazon S3 metric `BucketSizeBytes` includes the dimensions `BucketName` and `StorageType`, so you must specify both dimensions with get-metric-statistics.

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --start-time
 2017-01-23T14:23:00Z --end-time 2017-01-26T19:30:00Z --period 3600 --namespace
 AWS/S3 --statistics Maximum --dimensions Name=BucketName,Value=MyBucketName
 Name=StorageType,Value=StandardStorage --output table
```

You can see what dimensions are defined for a metric by using the list-metrics command.

Publish Single Data Points

To publish a single data point for a new or existing metric, use the put-metric-data command with one value and time stamp. For example, the following actions each publish one data point:

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --value 2
 --timestamp 2016-10-20T12:00:00.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --value 4
 --timestamp 2016-10-20T12:00:01.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --value 5
 --timestamp 2016-10-20T12:00:02.000Z
```

If you call this command with a new metric name, CloudWatch creates a metric for you. Otherwise, CloudWatch associates your data with the existing metric that you specified.

> **Note**
> When you create a metric, it can take up to two minutes before you can retrieve statistics for the new metric using the get-metric-statistics command. However, it can take up to fifteen minutes before the new metric appears in the list of metrics retrieved using the list-metrics command.

Although you can publish data points with time stamps as granular as one-thousandth of a second, CloudWatch aggregates the data to a minimum granularity of one minute. CloudWatch records the average (sum of all items divided by number of items) of the values received for every 1-minute period, as well as the number of samples, maximum value, and minimum value for the same time period. For example, the `PageViewCount` metric from the previous examples contains three data points with time stamps just seconds apart. CloudWatch aggregates the three data points because they all have time stamps within a one-minute period.

CloudWatch uses one-minute boundaries when aggregating data points. For example, CloudWatch aggregates the data points from the previous example because all three data points fall within the one-minute period that begins at `2016-10-20T12:00:00.000Z` and ends at `2016-10-20T12:01:00.000Z`.

You can use the **get-metric-statistics** command to retrieve statistics based on the data points that you published.

```
aws cloudwatch get-metric-statistics --namespace MyService --metric-name PageViewCount \
--statistics "Sum" "Maximum" "Minimum" "Average" "SampleCount" \
--start-time 2016-10-20T12:00:00.000Z --end-time 2016-10-20T12:05:00.000Z --period 60
```

The following is example output:

```
{
    "Datapoints": [
        {
            "SampleCount": 3.0,
            "Timestamp": "2016-10-20T12:00:00Z",
            "Average": 3.6666666666666665,
            "Maximum": 5.0,
            "Minimum": 2.0,
            "Sum": 11.0,
            "Unit": "None"
        }
    ],
    "Label": "PageViewCount"
}
```

Publish Statistic Sets

You can aggregate your data before you publish to CloudWatch. When you have multiple data points per minute, aggregating data minimizes the number of calls to **put-metric-data**. For example, instead of calling **put-metric-data** multiple times for three data points that are within three seconds of each other, you can aggregate the data into a statistic set that you publish with one call, using the `--statistic-values` parameter:

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService
 --statistic-values Sum=11,Minimum=2,Maximum=5,SampleCount=3 --
timestamp 2016-10-14T12:00:00.000Z
```

CloudWatch needs raw data points to calculate percentiles. If you publish data using a statistic set instead, you cannot retrieve percentile statistics for this data unless one of the following conditions is true:

- The SampleCount of the statistic set is 1.
- The Min and the Max of the statistic set are equal.

Publish the Value Zero

When your data is more sporadic and you have periods that have no associated data, you can choose to publish the value zero (0) for that period or no value at all. If you use periodic calls to `PutMetricData` to monitor the health of your application, you might want to publish zero instead of no value. For example, you can set a CloudWatch alarm to notify you if your application fails to publish metrics every five minutes. You want such an application to publish zeros for periods with no associated data.

You might also publish zeros if you want to track the total number of data points or if you want statistics such as minimum and average to include data points with the value 0.

Use Metric Math

Metric math enables you to query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics. You can visualize the resulting time series in the CloudWatch console

and add them to dashboards. For an example using AWS Lambda metrics, you could divide the **Errors** metric by the **Invocations** metric to get an error rate, and add the resulting time series to a graph on your CloudWatch dashboard.

You can also perform metric math programmatically, using the `GetMetricData` API operation.

Adding a Math Expression to a CloudWatch Graph

You can add a math expression to a graph on your CloudWatch dashboard. Each graph is limited to a maximum of 100 metrics and expressions, so you can add a math expression only if the graph has 99 or fewer metrics.

To add a math expression to a graph

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Create or edit a graph or line widget.
3. Choose **Graphed metrics**.
4. Choose **Add a math expression**. A new line appears for the expression.
5. For the **Details** column, type the math expression. The tables in the following section list the functions you can use in the expression.

   To use a metric or the result of another expression as part of the formula for this expression, use the value shown in the **Id** column. For example, **m1+m2** or **e1-MIN(e1)**.

   You can change the value of **Id**. It can include numbers, letters, and underscore, and must start with a lowercase letter. Changing the value of **Id** to a more meaningful name can also make a graph easier to understand. For example, changing from **m1** and **m2** to **errors** and **requests**.
6. For the **Label** column of the expression, type a name that describes what the expression is calculating.

   If the result of an expression is an array of time series, each of those time series is displayed on the graph with a separate line, with different colors. Immediately under the graph is a legend for each line in the graph. For a single expression that produces multiple time series, the legend captions for those time series are in the format *Expression-Label Metric-Label*. For example, if the graph includes a metric with a label of **Errors** and an expression **FILL(METRICS(), 0)** that has a label of **Filled With 0:**, one line in the legend would be **Filled With 0: Errors**. You can set *Expression-Label* to be empty to have the legend show only the original metric labels.

   When one expression produces an array of time series on the graph, you cannot change the colors used for each of those time series.
7. After you have added the desired expressions, you can optionally simplify the graph by hiding some of the original metrics. To hide a metric or expression, clear the check box to the left of the **Id** field.

Metric Math Syntax and Functions

The following sections explain the functions available for metric math. All functions must be written in uppercase letters (such as **AVG**), while the **Id** field for all metrics and math expressions must start with a lowercase letter.

The final result of any math expression must be a single time series or an array of time series. Some functions produce a scalar number. You can use these functions within a larger function that ultimately produces a time series. For example, taking the **AVG** of a single time series produces a scalar number, so it cannot be the final expression result. But you could use it in the function **m1-AVG(m1)** to display a time series of the difference between each individual data point and the average value of that data point.

Data Type Abbreviations

Some functions are valid for only certain types of data. The abbreviations in the following list are used in the tables of functions to represent the types of data supported for each function.

- **S** represents a scalar number, such as 2, -5, or 50.25.
- **TS** is a time series (a series of values for a single CloudWatch metric over time). For example, the **CPUUtilization** metric for instance `i-1234567890abcdef0` over the last three days.
- **TS[]** is an array of time series, such as the time series for multiple metrics.

The METRICS() Function

The **METRICS()** function returns all the metrics in the request. Math expressions are not included.

You can use **METRICS()** within a larger expression that produces a single time series or an array of time series. For example, the expression **SUM(METRICS())** returns a time series (TS) that is the sum of the values of all the graphed metrics. **METRICS()/100** returns an array of time series, each of which is a time series showing each data point of one of the metrics divided by 100.

You can use the **METRICS()** function with a string to return only the graphed metrics that contain that string in their **Id** field. For example, the expression **SUM(METRICS("errors"))** returns a time series that is the sum of the values of all the graphed metrics that have 'errors' in their **Id** field. You can also use **SUM([METRICS("4xx"), METRICS("5xx")])** to match multiple strings.

Basic Arithmetic Functions

The following table lists the basic arithmetic functions that are supported. Missing values in time series are treated as 0. If the value of a data point causes a function to attempt to divide by zero, the data point is dropped.

Operation	Arguments	Examples
Arithmetic operators: + - * / ^	S, S	PERIOD(m1)/60
	S, TS	5 * m1
	TS, TS	m1 - m2
	S, TS[]	SUM(100/[m1, m2])
	TS, TS[]	AVG([m1,m2]/m3)
		METRICS()*100
Unary subtraction -	S	-5*m1
	TS	-m1
	TS[]	SUM(-[m1, m2])

Functions Supported for Metric Math

The following table describes the functions you can use in math expressions. Write all functions in uppercase letters.

The final result of any math expression must be a single time series or an array of time series. Some functions in tables in the following sections produce a scalar number. You can use these functions within a larger function that ultimately produces a time series. For example, taking the **AVG** of a single time series produces a scalar number, so it cannot be the final expression result. But you could use it in the function **m1-AVG(m1)** to display a time series of the difference between each individual data point and the average value of that data point.

In the following table, every example in the **Examples** column is an expression that results in a single time series or an array of time series. This shows how functions that return scalar numbers can be used as part of a valid expression that produces a single time series.

Function	Arguments	Return Type*	Description	Examples
ABS	TS TS[]	TS TS[]	Returns the absolute value of each data point.	ABS(m1-m2) MIN(ABS([m1, m2])) ABS(METRICS())
AVG	TS TS[]	S TS	The **AVG** of a single time series returns a scalar representing the average of all the data points in the metric. The **AVG** of an array of time series returns a single time series. Missing values are treated as 0.	SUM([m1,m2])/ AVG(m2) AVG(METRICS())
CEIL	TS TS[]	TS TS[]	Returns the ceiling of each metric (the smallest integer greater than or equal to each value).	CEIL(m1) CEIL(METRICS()) SUM(CEIL(METRICS()))
FILL	TS, TS/S TS[], TS/S	TS TS[]	Fills the missing values of a metric with the specified filler value, when the metric values are sparse.	FILL(m1,10) FILL(METRICS(), 0) FILL(m1, MIN(m1))
FLOOR	TS TS[]	TS TS[]	Returns the floor of each metric (the largest integer less than or equal to each value).	FLOOR(m1) FLOOR(METRICS())
MAX	TS TS[]	S TS	The **MAX** of a single time series returns a scalar representing the maximum value of all data points in the metric. The **MAX** value of an array of time series returns a single time series.	MAX(m1)/m1 MAX(METRICS())
METRIC_COUNT	TS[]	S	Returns the number of metrics in the time series array.	m1/ METRIC_COUNT(METRICS())
METRICS()	null string	TS[]	The **METRICS()** function returns all the CloudWatch metrics in the request. Math expressions are not included You can use **METRICS()** within a larger expression that produces a	AVG(METRICS()) SUM(METRICS("errors"))

Function	Arguments	Return Type*	Description	Examples
			single time series or an array of time series. You can use the **METRICS()** function with a string to return only the graphed metrics that contain that string in their **Id** field. For example, the expression SUM(METRICS("errors")) returns a time series that is the sum of the values of all the graphed metrics that have 'errors' in their **Id** field. You can also use SUM([METRICS("4xx"), METRICS("5xx")]) to match multiple strings.	
MIN	TS TS[]	S TS	The **MIN** of a single time series returns a scalar representing the minimum value of all data points in the metric. The **MIN** of an array of time series returns a single time series.	m1-MIN(m1) MIN(METRICS())
PERIOD	TS	S	Returns the period of the metric in seconds. Valid input is metrics, not the results of other expressions.	m1/PERIOD(m1)
RATE	TS TS[]	TS TS[]	Returns the rate of change of the metric, per second. This is calculated as the difference between the latest data point value and the previous data point value, divided by the time difference in seconds between the two values.	RATE(m1) RATE(METRICS())
STDDEV	TS TS[]	S TS	The **STDDEV** of a single time series returns a scalar representing the standard deviation of all data points in the metric. The **STDDEV** of an array of time series returns a single time series.	m1/STDDEV(m1) STDDEV(METRICS())
SUM	TS TS[]	S TS	The **SUM** of a single time series returns a scalar representing the sum of the values of all data points in the metric. The **SUM** of an array of time series returns a single time series.	SUM(METRICS())/ SUM(m1) SUM([m1,m2]) SUM(METRICS("errors"))/ SUM(METRICS("requests"))*100

*Using only a function that returns a scalar number is not valid, as all final results of expressions must be a single time series or an array of time series. Instead, use these functions as part of a larger expression that returns a time series.

Using Metric Math with the GetMetricData API Operation

You can use **GetMetricData** to perform calculations using math expressions, as well as to retrieve large batches of metric data in one API call. For more information, see GetMetricData.

Collect Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent

The unified CloudWatch agent enables you to do the following:

- Collect more system-level metrics from Amazon EC2 instances, including in-guest metrics, in addition to the metrics listed in Amazon EC2 Metrics and Dimensions (p. 152). The additional metrics are listed in Metrics Collected by the CloudWatch Agent (p. 125).
- Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
- Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.
- Retrieve custom metrics from your applications or services using the StatsD and collectd protocols. StatsD is supported on both Linux servers and servers running Windows Server. collectd is supported only on Linux servers.

You can store and view the metrics you collect with the CloudWatch agent in CloudWatch just as you can with any other CloudWatch metrics. The default namespace for metrics collected by the CloudWatch agent is `CWAgent`, although you can specify a different namespace when you configure the agent.

The logs collected by the unified CloudWatch agent are processed and stored in CloudWatch Logs, just like logs collected by the older CloudWatch Logs agent. For information about CloudWatch Logs pricing, see Amazon CloudWatch Pricing.

Metrics collected by the CloudWatch agent are billed as custom metrics. For more information about CloudWatch metrics pricing, see Amazon CloudWatch Pricing.

The steps in this section explain how to install the unified CloudWatch agent on Amazon EC2 instances and on-premises servers. For more information about the metrics that can be collected by the CloudWatch agent, see Metrics Collected by the CloudWatch Agent (p. 125).

Supported Operating Systems

The CloudWatch agent is supported on the following operating systems:

- Amazon Linux version 2014.03.02 or later
- Amazon Linux 2
- Ubuntu Server version 16.04 and 14.04
- CentOS version 7.0 and 6.5
- Red Hat Enterprise Linux (RHEL) version 7.5, 7.4, 7.0, and 6.5
- Debian 8.0
- SUSE Linux Enterprise Server (SLES) 12 or later
- 64-bit versions of Windows Server 2016, Windows Server 2012, and Windows Server 2008.

Installation Process Overview

The general flow of installing the CloudWatch agent is as follows:

1. Create the IAM roles and users that you need for the CloudWatch agent. They enable CloudWatch to collect metrics from the server, and to integrate with AWS Systems Manager.
2. If you are installing on an Amazon EC2 instance, attach an IAM role to the instance. If you are installing on an on-premises server, create an IAM user to enable the CloudWatch agent to write information to CloudWatch.
3. Download the agent package, using either AWS Systems Manager Run Command or a public Amazon S3 download link.
4. Modify the CloudWatch agent configuration files, and create a named profile for the CloudWatch agent. Creating the named profile is optional when installing the agent on an Amazon EC2 instance.
5. Start the agent, using either Systems Manager Run Command or the command line.

Contents

Create IAM Roles and Users for Use With CloudWatch Agent

Access to AWS resources requires permissions. You can create IAM roles and users that include the permissions you need for the CloudWatch agent to write metrics to CloudWatch, and for the CloudWatch agent to communicate with Amazon EC2 and AWS Systems Manager. You use IAM roles on Amazon EC2 instances, and you use IAM users with on-premises servers to enable the agent to send data to CloudWatch.

One role and one user enable CloudWatch agent to be installed on a server and send metrics to CloudWatch. The other role or user is needed to store your CloudWatch agent configuration in Systems Manager Parameter Store, which enables multiple servers to use one CloudWatch agent configuration.

The ability to write to Parameter Store is a broad and powerful permission, and should be used only when you need it, and should not be attached to multiple instances in your deployment. If you are going to store your CloudWatch agent configuration in Parameter Store, you should set up one instance where you perform this configuration, and use the IAM role with permissions to write to Parameter Store only on this instance, and only while you are working with and saving the CloudWatch agent configuration file.

> **Note**
> We recently modified the following procedures by using new **CloudWatchAgentServerPolicy** and **CloudWatchAgentAdminPolicy** policies created by Amazon, instead of requiring customers to create these policies themselves. For writing files to and downloading files from the Parameter Store, the policies created by Amazon support only files with names that start with "AmazonCloudWatch-". If you have a CloudWatch agent configuration file with a filename that does not start with `AmazonCloudWatch-`, these policies cannot be used to write the file to Parameter Store or download it from Parameter Store.

Create IAM Roles to Use with CloudWatch Agent on Amazon EC2 Instances

The first procedure creates the IAM role that you must attach to each Amazon EC2 instance that runs the CloudWatch agent. This role provides permissions for reading information from the instance and writing it to CloudWatch.

The second procedure creates the IAM role that you must attach to the Amazon EC2 instance being used to create the CloudWatch agent configuration file, if you are going to store this file in Systems Manager Parameter Store so that other servers can use it. This role provides permissions for writing to Parameter Store, in addition to the permissions for reading information from the instance and writing it to CloudWatch. This role includes permissions sufficient to run the CloudWatch agent as well as to write to Parameter Store.

To create the IAM role necessary for each server to run CloudWatch agent

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose **Roles**, **Create role**.

3. For **Choose the service that will use this role**, choose **EC2 Allows EC2 instances to call AWS services on your behalf**. Choose **Next: Permissions**.

4. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. Use the search box to find the policy, if necessary.

5. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. Use the search box to find the policy, if necessary. This policy is not necessary if you start and configure the agent only through the command line.

6. Choose **Next: Review**.

7. Confirm that **CloudWatchAgentServerPolicy** and optionally **AmazonEC2RoleforSSM** appear next to **Policies**. In **Role name**, type a name for the role, such as CloudWatchAgentServerRole. Optionally give it a description, and choose **Create role**.

   The role is now created.

The following procedure creates the IAM role that can also write to Parameter Store. You need to use this role if you are going to store the agent configuration file in Parameter Store so that other servers can use it. This role provides permissions for writing to Parameter Store, in addition to the permissions for reading information from the instance and writing it to CloudWatch. The permissions for writing to Parameter Store provide broad powers, and should not be attached to all your servers, and should be used only by administrators. After you are finished creating the agent configuration file and copying it to Parameter Store, you should detach this role from the instance and use the **CloudWatchAgentServerPolicy** instead.

To create the IAM role necessary for an administrator to save an agent configuration file to Systems Manager Parameter Store

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose **Roles**, **Create role**.

3. For **Choose the service that will use this role**, choose **EC2 Allows EC2 instances to call AWS services on your behalf**. Choose **Next: Permissions**.

4. In the list of policies, select the check box next to **CloudWatchAgentAdminPolicy**. Use the search box to find the policy, if necessary.

Amazon CloudWatch User Guide
Create IAM Users to Use with CloudWatch
Agent on On-premises Servers

5. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. Use the search box to find the policy, if necessary. This policy is not necessary if you start and configure the agent only through the command line.

6. Choose **Next: Review**.

7. Confirm that **CloudWatchAgentAdminPolicy** and optionally **AmazonEC2RoleforSSM** appear next to **Policies**. In **Role name**, type a name for the role, such as CloudWatchAgentAdminRole. Optionally give it a description, and choose **Create role**.

   The role is now created.

Create IAM Users to Use with CloudWatch Agent on On-premises Servers

The first procedure creates the IAM user that you need for running the CloudWatch agent. This user provides permissions for sending data to CloudWatch.

The second procedure creates the IAM user that you can use when creating the CloudWatch agent configuration file, if you are going to store this file in Systems Manager Parameter Store so that other servers can use it. This user provides permissions for writing to Parameter Store, in addition to the permissions for writing data to CloudWatch.

To create the IAM user necessary for CloudWatch agent to write data to CloudWatch

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose **Users**, **Add user**.

3. Type the user name for the new user.

4. Select **Programmatic access**, and choose **Next: Permissions**.

5. Choose **Attach existing policies directly**.

6. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. Use the search box to find the policy, if necessary.

7. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. Use the search box to find the policy, if necessary. This policy is not necessary if you start and configure the agent only through the command line.

8. Choose **Next: Review**.

9. Confirm that the correct policies are listed, and choose **Create user**.

10. Next to the name of the new user, choose **Show**. Copy the access key and secret key to a file so that you can use them when installing the agent, and choose **Close**.

The following procedure creates the IAM user that can also write to Parameter Store. If you are going to store the agent configuration file in Parameter Store so that other servers can use it, you need to use this user. This user provides permissions for writing to Parameter Store, in addition to the permissions for reading information from the instance and writing it to CloudWatch. The permissions for writing to Systems Manager Parameter Store provide broad powers, and should not be attached to all your servers, and should be used only by administrators. You should use this IAM user only when you are storing the agent configuration file in Parameter Store.

To create the IAM user necessary to store the configuration file in Parameter Store and send information to CloudWatch

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose **Users**, **Add user**.

3. Type the user name for the new user.

4. Select **Programmatic access**, and choose **Next: Permissions**.

5. Choose **Attach existing policies directly**.

6. In the list of policies, select the check box next to **CloudWatchAgentAdminPolicy**. Use the search box to find the policy, if necessary.

7. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. Use the search box to find the policy, if necessary. This policy is not necessary if you start and configure the agent only through the command line.

8. Choose **Next: Review**.

9. Confirm that the correct policies are listed, and choose **Create user**.

10. Next to the name of the new user, choose **Show**. Copy the access key and secret key to a file so that you can use them when installing the agent, and choose **Close**.

Install the CloudWatch Agent on an Amazon EC2 Instance

When you first start using CloudWatch agent, you download it to a server and configure the agent. You can then use the agent with that configuration directly on that server, and if you save the configuration in AWS Systems Manager Parameter Store, you can also use the same configuration when you install the CloudWatch agent on other servers.

Topics

Getting Started: Installing the CloudWatch Agent on Your First Instance

To download and install the CloudWatch agent on a running Amazon EC2 instance, you can use either AWS Systems Manager or the command line. With either method, you must first create an IAM role and attach it to the instance.

Topics

Attach an IAM Role to the Instance

An IAM role for the instance profile is required when you install the CloudWatch agent on an Amazon EC2 instance. This role enables the CloudWatch agent to perform actions on the instance. Use one of the roles you created earlier. For more information about creating these roles, see Create IAM Roles and

Users for Use With CloudWatch Agent (p. 54). You can scroll through the list to find them, or use the search box.

If you are going to use this instance to create the CloudWatch agent configuration file and copy it to Systems Manager Parameter Store, use the role you created that has permissions to write to Parameter Store. This role may be called **CloudWatchAgentAdminRole**.

For all other instances, select the role that includes just the permissions needed to install and run the agent. This role may be called **CloudWatchAgentServerRole**.

Attach this role to the instance on which you install the CloudWatch agent. For more information, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide for Windows Instances*.

Download the CloudWatch Agent Package on an Amazon EC2 Instance

You can download the CloudWatch agent package using either Systems Manager Run Command or an Amazon S3 download link.

Download the CloudWatch Agent on an Amazon EC2 Instance Using AWS Systems Manager

Before you can use Systems Manager to install the CloudWatch agent, you must make sure that the instance is configured correctly for Systems Manager.

Install or Update the SSM Agent

On an Amazon EC2 instance, the CloudWatch agent requires that the instance is running version 2.2.93.0 or later. Before you install the CloudWatch agent, update or install the SSM Agent on the instance if you haven't already done so.

For information about installing or updating the SSM Agent on an instance running Linux, see Installing and Configuring the SSM Agent on Linux Instances in the *AWS Systems Manager User Guide*.

For information about installing or updating the SSM Agent, see Installing and Configuring the SSM Agent in the *AWS Systems Manager User Guide*.

(Optional) Verify Systems Manager Prerequisites

Before you use Systems Manager Run Command to install and configure the CloudWatch agent, verify that your instances meet the minimum Systems Manager requirements. For more information, see Systems Manager Prerequisites in the *AWS Systems Manager User Guide*.

Verify Internet Access

Your Amazon EC2 instances must have outbound internet access in order to send data to CloudWatch or CloudWatch Logs. For more information about how to configure internet access, see Internet Gateways in the *Amazon VPC User Guide*.

The endpoints and ports to configure on your proxy are as follows:

- If you are using the agent to collect metrics, you must whitelist the CloudWatch endpoints for the appropriate regions. These endpoints are listed in Amazon CloudWatch in the *Amazon Web Services General Reference*.
- If you are using the agent to collect logs, you must whitelist the CloudWatch Logs endpoints for the appropriate regions. These endpoints are listed in Amazon CloudWatch Logs in the *Amazon Web Services General Reference*.

- If you are using SSM to install the agent or Parameter Store to store your configuration file, you must whitelist the SSM endpoints for the appropriate regions. These endpoints are listed in AWS Systems Manager in the *Amazon Web Services General Reference*.

Download the CloudWatch Agent Package

Systems Manager Run Command enables you to manage the configuration of your instances. You specify a Systems Manager document, specify parameters, and execute the command on one or more instances. The SSM Agent on the instance processes the command and configures the instance as specified.

To download the CloudWatch agent using Systems Manager

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.

2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.

3. Choose **Run command**.

4. In the **Command document** list, choose **AWS-ConfigureAWSPackage**.

5. In the **Targets** area, choose the instance on which to install the CloudWatch agent. If you do not see a specific instance, it might not be configured for Run Command. For more information, see Systems Manager Prerequisites in the *Amazon EC2 User Guide for Windows Instances*.

6. In the **Action** list, choose **Install**.

7. In the **Name** field, type **AmazonCloudWatchAgent**.

8. Leave **Version** set to **latest** to install the latest version of the agent.

9. Choose **Run**.

10. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully installed.

Download the CloudWatch Agent Package on an Amazon EC2 Instance Using an S3 Download Link

You can use an Amazon S3 download link to download the CloudWatch agent package on an Amazon EC2 instance server. Choose the download link from this table, depending on your architecture and platform.

Arch	Platform	Download Link	Signature File Link
amd64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Centos	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/	https://s3.amazonaws.com/amazoncloudwatch-agent/

Arch	Platform	Download Link	Signature File Link
		redhat/amd64/latest/amazon-cloudwatch-agent.rpm	redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Debian	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Windows	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/AmazonCloudWatchAgent.zip	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/AmazonCloudWatchAgent.zip.sig

To use the command line to install the CloudWatch agent on an Amazon EC2 instance

1. Make a directory for downloading and unzipping the agent package. For example, `tmp/AmazonCloudWatchAgent`. Then change into that directory.

2. Download the CloudWatch agent. Use a download link from the previous table. For a Linux server, type the following:

```
wget download-link
```

For a server running Windows Server, download the following file:

```
https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/
AmazonCloudWatchAgent.zip
```

3. After you have downloaded the package, you can optionally use a GPG signature file to verify the package signature. For more information, see Verify the Signature of the CloudWatch Agent Package (p. 74).

4. If you downloaded a .zip package, unzip the package.

```
unzip AmazonCloudWatchAgent.zip
```

5. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and type:

```
rpm -U ./amazon-cloudwatch-agent.rpm
```

If you downloaded a DEB package on a Linux server, change to the directory containing the package and type:

```
dpkg -i -E ./amazon-cloudwatch-agent.deb
```

On a server running Windows Server, open PowerShell, change to the directory containing the
unzipped package, and use the `install.ps1` script to install it.

(Optional) Modify the Common Configuration and Named Profile for CloudWatch Agent

The CloudWatch agent package you have downloaded includes a configuration file called `common-config.toml`. You can use this file to specify proxy, credential, and region information. On a server
running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server
running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent`
directory.

The default `common-config.toml` is as follows:

When you install the CloudWatch agent on an Amazon EC2 instance, modify this file only if you need to
specify proxy settings or if the agent should send metrics to CloudWatch in a different region than where
the instance is located.

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##              Instance role is used for EC2 case by default.
##              AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

All lines are commented out initially. To set the credential profile or proxy settings, remove the # from
that line and specify a value. You can edit this file manually, or by using the `RunShellScript` Run
Command in Systems Manager:

- **proxy settings** If your servers use HTTP or HTTPS proxies to contact AWS services, specify those
  proxies in the `http_proxy` and `https_proxy` fields. If there are URLs that should be excluded from
  proxying, specify them in the `no_proxy` field, separated by commas.

- **shared_credential_profile** To have the CloudWatch agent send metrics to CloudWatch in the same
  region where the instance is located, you don't need to modify this line if you have attached an IAM
  role with the proper permissions to the instance, and you don't need to use the `aws configure`
  command to create a named profile for the agent.

  Otherwise, you can use this line to specify the named profile that CloudWatch agent is to use in the
  AWS config file. If you do so, CloudWatch agent uses the region settings in that named profile.

- **shared_credential_file** Use this line to specify a path to a file containing credentials to use, if you
  don't want to use the default path.

After modifying `common-config.toml`, if you need to specify region information for the CloudWatch agent, create a named profile for the CloudWatch agent in the AWS config file. When you create this profile, do so as the root or administrator.

Following is an example of the profile for the configuration file:

```
[AmazonCloudWatchAgent]
region=us-west-1
```

To be able to send the CloudWatch data to a different region, make sure the IAM role that you attached to this instance has permissions to write the CloudWatch data in that region.

Following is an example of using the `aws configure` command to create a named profile for the CloudWatch agent. This example assumes that you are using the default profile name of `AmazonCloudWatchAgent`.

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

- Type the following command and follow the prompts:

```
aws configure --profile AmazonCloudWatchAgent
```

Create the Agent Configuration File on Your First Instance

After you have downloaded the CloudWatch agent, you must create the configuration file before you start the agent on any servers. For more information, see Create the CloudWatch Agent Configuration File (p. 81).

Start the CloudWatch Agent

To start the agent on the same server where you created the agent configuration file, follow these steps. To use this configuration file on other servers, see Installing CloudWatch Agent on Additional Instances Using Your Agent Configuration (p. 63).

Start the CloudWatch Agent Using Run Command

Follow these steps to start the agent using Systems Manager Run Command.

To start the CloudWatch agent using Run Command

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManageAgent**.
5. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
6. In the **Action** list, choose **configure**.
7. In the **Optional Configuration Source** list, choose **ssm**.
8. In the **Optional Configuration Location** box, type the name of the agent configuration file that you created and saved to Systems Manager Parameter Store, as explained in Create the CloudWatch Agent Configuration File (p. 81).

9. In the **Optional Restart** list, choose **yes** to start the agent after you have finished these steps.

10. Choose **Run**.

11. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully started.

Start the CloudWatch Agent on an Amazon EC2 Instance Using the Command Line

Follow these steps to use the command line to install the CloudWatch agent on an Amazon EC2 instance.

To use the command line to start the CloudWatch agent on an Amazon EC2 instance

*   In this command, `-a fetch-config` causes the agent to load the latest version of the CloudWatch agent configuration file, and `-s` starts the agent.

    Linux: type the following if you saved the configuration file in the Systems Manager Parameter Store:

    ```
    sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
    m ec2 -c ssm:configuration-parameter-store-name -s
    ```

    Linux: type the following if you saved the configuration file on the local computer:

    ```
    sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
    m ec2 -c file:configuration-file-path -s
    ```

    Windows Server: if you saved the agent configuration file in Systems Manager Parameter Store, use the following command. From the PowerShell console, type the following:

    ```
    ./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c ssm:configuration-
    parameter-store-name -s
    ```

    Windows Server: if you saved the agent configuration file on the local computer, use the following command. From the PowerShell console, type the following:

    ```
    ./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c file:configuration-file-
    path -s
    ```

Installing CloudWatch Agent on Additional Instances Using Your Agent Configuration

After you have a CloudWatch agent configuration saved in Parameter Store, you can use it when you install the agent on other servers.

Topics

*   Create an IAM Role for Systems Manager and the CloudWatch Agent (p. 64)
*   Download the CloudWatch Agent Package on an Amazon EC2 Instance (p. 64)
*   (Optional) Modify the Common Configuration and Named Profile for CloudWatch Agent (p. 66)
*   Start the CloudWatch Agent (p. 68)

Create an IAM Role for Systems Manager and the CloudWatch Agent

An IAM role for the instance profile is required when you install the CloudWatch agent on an Amazon EC2 instance. This role enables the CloudWatch agent to perform actions on the instance. Use the role you created earlier that includes just the permissions needed for installing and running the agent. This role may be called **CloudWatchAgentServerPolicy**.

Attach this role to the instance where you install the CloudWatch agent. For more information, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide for Windows Instances*.

Download the CloudWatch Agent Package on an Amazon EC2 Instance

You can download the CloudWatch agent package using either Systems Manager Run Command or an Amazon S3 download link.

Download the CloudWatch Agent on an Amazon EC2 Instance Using Systems Manager

Before you can use Systems Manager to install the CloudWatch agent, you must make sure that the instance is configured correctly for Systems Manager.

Install or Update the SSM Agent

On an Amazon EC2 instance, the CloudWatch agent requires that the instance is running version 2.2.93.0 or later. Before you install the CloudWatch agent, update or install the SSM Agent on the instance if you haven't already done so.

For information about installing or updating the SSM Agent on an instance running Linux, see Installing and Configuring the SSM Agent on Linux Instances in the *AWS Systems Manager User Guide*.

For information about installing or updating the SSM Agent, see Installing and Configuring SSM Agent in the *AWS Systems Manager User Guide*.

(Optional) Verify Systems Manager Prerequisites

Before you use Systems Manager Run Command to install and configure the CloudWatch agent, verify that your instances meet the minimum Systems Manager requirements. For more information, see Systems Manager Prerequisites in the *AWS Systems Manager User Guide*.

Verify Internet Access

Your Amazon EC2 instances must have outbound internet access in order to send data to CloudWatch or CloudWatch Logs. For more information about how to configure internet access, see Internet Gateways in the *Amazon VPC User Guide*.

Download the CloudWatch Agent Package

Systems Manager Run Command enables you to manage the configuration of your instances. You specify a Systems Manager document, specify parameters, and execute the command on one or more instances. The SSM Agent on the instance processes the command and configures the instance as specified.

To download the CloudWatch agent using Run Command

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

-or-

If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.

3. Choose **Run command**.

4. In the **Command document** list, choose **AWS-ConfigureAWSPackage**.

5. In the **Targets** area, choose the instance on which to install the CloudWatch agent. If you do not see a specific instance, it might not be configured for Run Command. For more information, see Systems Manager Prerequisites in the *Amazon EC2 User Guide for Windows Instances*.

6. In the **Action** list, choose **Install**.

7. In the **Name** box, type **AmazonCloudWatchAgent**.

8. Leave **Version** set to **latest** to install the latest version of the agent.

9. Choose **Run**.

10. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully installed.

Download the CloudWatch Agent Package on an Amazon EC2 Instance Using an S3 Download Link

You can use an Amazon S3 download link to download the CloudWatch agent package on an Amazon EC2 instance server. Choose the download link from this table, depending on your architecture and platform.

Arch	Platform	Download Link	Signature File Link
amd64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Centos	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Debian	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig

Arch	Platform	Download Link	Signature File Link
amd64	Windows	https://s3.amazonaws.com/ amazoncloudwatch-agent/ windows/amd64/latest/ AmazonCloudWatchAgent.zip	https://s3.amazonaws.com/ amazoncloudwatch-agent/ windows/amd64/latest/ AmazonCloudWatchAgent.zip.sig

To use the command line to install the CloudWatch agent on an Amazon EC2 instance

1. Make a directory for downloading and unzipping the agent package. For example, `tmp/ AmazonCloudWatchAgent`. Then change into that directory.

2. Download the CloudWatch agent. Use a download link from the previous table. For a Linux server, type the following:

    ```
    wget download-link
    ```

    For a server running Windows Server, download the following file:

    ```
    https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/
    AmazonCloudWatchAgent.zip
    ```

3. After you have downloaded the package, you can optionally use a GPG signature file to verify the package signature. For more information, see Verify the Signature of the CloudWatch Agent Package (p. 74).

4. Unzip the package.

    ```
    unzip AmazonCloudWatchAgent.zip
    ```

5. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and type:

    ```
    rpm -U ./amazon-cloudwatch-agent.rpm
    ```

    If you downloaded a DEB package on a Linux server, change to the directory containing the package and type:

    ```
    dpkg -i -E ./amazon-cloudwatch-agent.deb
    ```

    On a server running Windows Server, open PowerShell, change to the directory containing the unzipped package, and use the `install.ps1` script to install it.

(Optional) Modify the Common Configuration and Named Profile for CloudWatch Agent

The CloudWatch agent package you have downloaded includes a configuration file called `common-config.toml`. You can use this file to specify proxy, credential, and region information. On a server running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent` directory.

The default `common-config.toml` is as follows:

When you install the CloudWatch agent on an Amazon EC2 instance, you need to modify this file only if you need to specify proxy settings or if the agent should send metrics to CloudWatch in a different region than where the instance is located.

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##              Instance role is used for EC2 case by default.
##              AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

All lines are commented out initially. To set the credential profile or proxy settings, remove the # from that line and specify a value. You can edit this file manually, or by using the `RunShellScript` Run Command in Systems Manager:

- **shared_credential_profile** To have the CloudWatch agent send metrics to CloudWatch in the same region where the instance is located, you don't need to modify this line if you have attached an IAM role with the proper permissions to the instance. You also don't need to use the `aws configure` command to create a named profile for the agent.

  Otherwise, you can use this line to specify the named profile that CloudWatch agent is to use in the AWS credentials and AWS config files. If you do so, CloudWatch agent uses the credential and the region settings in that named profile.
- **shared_credential_file** Use this line to specify a path to a file containing credentials to use, if you don't want to use the default path.
- **proxy settings** If your servers use HTTP or HTTPS proxies to contact AWS services, specify those proxies in the `http_proxy` and `https_proxy` fields. If there are URLs that should be excluded from proxying, specify them in the `no_proxy` field, separated by commas.

After modifying `common-config.toml`, if you need to specify credential and region information for the CloudWatch agent, create a named profile for the CloudWatch agent in the AWS credentials and AWS config files. When you create this profile, do so as the root or administrator. Following is an example of this profile in the credentials file:

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
```

For `my_access_key` and `my_secret_key`, use the keys from the IAM user that does not have the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for the CloudWatch agent, see Create IAM Users to Use with CloudWatch Agent on On-premises Servers (p. 56).

Following is an example of the profile for the configuration file:

```
[AmazonCloudWatchAgent]
region=us-west-1
```

Following is an example of using the `aws configure` command to create a named profile for the CloudWatch agent. This example assumes that you are using the default profile name of `AmazonCloudWatchAgent`.

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

- Type the following command and follow the prompts:

```
aws configure --profile AmazonCloudWatchAgent
```

Start the CloudWatch Agent

You can start the agent using Systems Manager Run Command or the command line.

Start the CloudWatch Agent Using Systems Manager Run Command

Follow these steps to start the agent using Systems Manager Run Command.

To start the CloudWatch agent using Run Command

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManageAgent**.
5. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
6. In the **Action** list, choose **configure**.
7. In the **Optional Configuration Source** list, choose **ssm**.
8. In the **Optional Configuration Location** box, type the name of the agent configuration file that you created and saved to Systems Manager Parameter Store, as explained in Create the CloudWatch Agent Configuration File (p. 81).
9. In the **Optional Restart** list, choose **yes** to start the agent after you have finished these steps.
10. Choose **Run**.
11. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully started.

Start the CloudWatch Agent on an Amazon EC2 Instance Using the Command Line

Follow these steps to use the command line to install the CloudWatch agent on an Amazon EC2 instance.

To use the command line to start the CloudWatch agent on an Amazon EC2 instance

- In this command, `-a fetch-config` causes the agent to load the latest version of the CloudWatch agent configuration file, and `-s` starts the agent.

Linux: type the following if you saved the configuration file in the Systems Manager Parameter Store:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
m ec2 -c ssm:configuration-parameter-store-name -s
```

Linux: type the following if you saved the configuration file on the local computer:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
m ec2 -c file:configuration-file-path -s
```

Windows Server: if you saved the agent configuration file in Systems Manager Parameter Store, use the following command. From the PowerShell console, type the following:

```
./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c ssm:configuration-
parameter-store-name -s
```

Windows Server: if you saved the agent configuration file on the local computer, use the following command. From the PowerShell console, type the following:

```
./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c file:configuration-file-
path -s
```

Install the CloudWatch Agent on an On-Premises Server

When you first start using CloudWatch agent, you download it to a server and configure the agent. You can then use the agent with that configuration directly on that server. If you save the configuration in AWS Systems Manager Parameter Store, you can also use the same configuration when you install the CloudWatch agent on other servers.

Topics
- Getting Started: Installing the CloudWatch Agent on Your First Server (p. 69)
- Installing CloudWatch Agent on Additional Servers Using Your Agent Configuration (p. 77)

Getting Started: Installing the CloudWatch Agent on Your First Server

To download and install the CloudWatch agent on an on-premises server, you can use either AWS Systems Manager or the command line.

With either method, you must first create an IAM user with permissions to write to CloudWatch.

Topics
- Download the CloudWatch Agent on an On-Premises Server (p. 70)
- Modify the Common Configuration and Named Profile for CloudWatch Agent (p. 72)
- Create the CloudWatch Agent Configuration File (p. 73)
- Start the CloudWatch Agent (p. 73)

- Verify the Signature of the CloudWatch Agent Package (p. 74)

Download the CloudWatch Agent on an On-Premises Server

To download the CloudWatch agent, you can use Systems Manager or the command line.

Download Using Systems Manager

To use Systems Manager Run Command, you must register your on-premises server with Amazon EC2 Systems Manager. For more information, see Setting Up Systems Manager in Hybrid Environments in the *AWS Systems Manager User Guide*.

If you have already registered your server, update your SSM Agent to the latest version.

For information about updating the SSM Agent on a server running Linux, see Install the SSM Agent on Servers and VMs in Your Linux Hybrid Environment in the *AWS Systems Manager User Guide*.

For information about updating the SSM Agent on a server running Windows Server, see Install the SSM Agent on Servers and VMs in Your Windows Hybrid Environment in the *AWS Systems Manager User Guide*.

To use the SSM Agent to download the CloudWatch agent package on an on-premises server

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, select the button next to **AWS-ConfigureAWSPackage**.
5. In the **Targets** area, select the server on which to install the CloudWatch agent. If you do not see a specific server, it might not be configured for Run Command. For more information, see Systems Manager Prerequisites in the *Amazon EC2 User Guide for Windows Instances*.
6. In the **Action** list, choose **Install**.
7. In the **Name** box, type **AmazonCloudWatchAgent**.
8. Leave **Version** blank to install the latest version of the agent.
9. Choose **Run**.

   The agent package is downloaded, and the next steps are to configure and start it.

Download Using an S3 Download link

If you are using the command line to download the agent, first choose the download link from this table. Choose the link depending on your architecture and platform.

Arch	Platform	Download Link	Signature File Link
amd64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/ amazoncloudwatch-agent/ amazon_linux/amd64/latest/ amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ amazon_linux/amd64/ latest/amazon-cloudwatch- agent.rpm.sig

Arch	Platform	Download Link	Signature File Link
amd64	Centos	https://s3.amazonaws.com/ amazoncloudwatch-agent/ centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Redhat	https://s3.amazonaws.com/ amazoncloudwatch-agent/ redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	SUSE	https://s3.amazonaws.com/ amazoncloudwatch-agent/ suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Debian	https://s3.amazonaws.com/ amazoncloudwatch-agent/ debian/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/ amazoncloudwatch-agent/ debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Ubuntu	https://s3.amazonaws.com/ amazoncloudwatch-agent/ ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/ amazoncloudwatch-agent/ ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Windows	https://s3.amazonaws.com/ amazoncloudwatch-agent/ windows/amd64/latest/ AmazonCloudWatchAgent.zip	https://s3.amazonaws.com/ amazoncloudwatch-agent/ windows/amd64/latest/ AmazonCloudWatchAgent.zip.sig

To use the command line to download the CloudWatch agent on an on-premises server

1. Make a directory for downloading and unzipping the agent package. For example, `tmp/ AmazonCloudWatchAgent`. Then change into that directory.

2. Download the CloudWatch agent. Use a download link from the previous table. For a Linux server, type the following:

```
wget download-link
```

For a server running Windows Server, download the following file:

```
https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/
AmazonCloudWatchAgent.zip
```

3. After you have downloaded the package, you can optionally use a GPG signature file to verify the package signature. For more information, see Verify the Signature of the CloudWatch Agent Package (p. 74).

4. Unzip the package.

```
unzip AmazonCloudWatchAgent.zip
```

5. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and type:

```
rpm -U ./amazon-cloudwatch-agent.rpm
```

If you downloaded a DEB package on a Linux server, change to the directory containing the package and type:

```
dpkg -i -E ./amazon-cloudwatch-agent.deb
```

On a server running Windows Server, open PowerShell, change to the directory containing the unzipped package, and use the `install.ps1` script to install it.

Modify the Common Configuration and Named Profile for CloudWatch Agent

The CloudWatch agent package you have downloaded includes a configuration file called `common-config.toml`. You can use this file to specify proxy, credential, and region information. On a server running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent` directory.

The default `common-config.toml` is as follows:

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##              Instance role is used for EC2 case by default.
##              AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

All lines are commented out initially. To set the credential profile or proxy settings, remove the # from that line and specify a value. You can edit this file manually, or by using the `RunShellScript` Run Command in Systems Manager:

- **shared_credential_profile** You can specify a name for the named profile that CloudWatch agent is to look for in the AWS credentials and AWS config files. If you don't specify a name here, the CloudWatch agent looks for the default profile name, `AmazonCloudWatchAgent`.

- **shared_credential_file** If you don't want to use the default path, use this line to specify a path to a file containing credentials to use.

- **proxy settings** If your servers use HTTP or HTTPS proxies to contact AWS services, specify those proxies in the `http_proxy` and `https_proxy` fields. If there are URLs that should be excluded from proxying, specify them in the `no_proxy` field, separated by commas.

After modifying `common-config.toml`, you should make sure the profile name that you specified, or the default profile name of `AmazonCloudWatchAgent`, exists in the root user's AWS credentials and config files. This profile is used for credential and region information during CloudWatch agent setup. Following is an example of this profile in the credentials file:

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
```

For `my_access_key` and `my_secret_key`, use the keys from the IAM user that does not have the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for the CloudWatch agent, see Create IAM Users to Use with CloudWatch Agent on On-premises Servers (p. 56).

Following is an example of the profile for the configuration file:

```
[AmazonCloudWatchAgent]
region=us-west-1
```

The named profile in the credentials file contains the credentials to be used for the CloudWatch agent. These credentials are used for permissions to write metric data to CloudWatch, and to download information from Systems Manager Parameter Store during CloudWatch agent installation. You can obtain the credentials to use for this section by creating an IAM user for the CloudWatch agent, as explained previously in this section.

The named profile in the configuration file specifies the region to which the CloudWatch metrics are published, when the CloudWatch agent runs on an on-premises server. When you use the `aws configure` command to modify the profile, do so as the root or administrator.

Following is an example of using the `aws configure` command to create a named profile for the CloudWatch agent. This example assumes that you are using the default profile name of `AmazonCloudWatchAgent`.

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

- Type the following command and follow the prompts:

```
aws configure --profile AmazonCloudWatchAgent
```

Create the CloudWatch Agent Configuration File

CloudWatch agent uses a configuration file to specify the metrics to be collected and other agent configuration data. You must customize this file before you start the agent. For more information, see Create the CloudWatch Agent Configuration File (p. 81).

Start the CloudWatch Agent

You can start the CloudWatch agent using either Systems Manager Run Command or the command line.

To use the SSM Agent to start the CloudWatch agent on an on-premises server

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

-or-

If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.

3. Choose **Run command**.
4. In the **Command document** list, select the button next to **AmazonCloudWatch-ManageAgent**.
5. In the **Targets** area, select the instance where you installed the agent.
6. In the **Action** list, choose **configure**.
7. In the **Mode** list, choose **onPremise**.
8. In the **Optional Configuration Location** box, type the name of the agent configuration file that you created with the wizard and stored in Parameter Store.
9. Choose **Run**.

The agent starts with the configuration that you specified in the configuration file.

To use the command line to start the CloudWatch agent on an on-premises server

- In this command, `-a fetch-config` causes the agent to load the latest version of the CloudWatch agent configuration file, and `-s` starts the agent.

  Linux: type the following if you saved the configuration file in the Systems Manager Parameter Store:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
  m onPremise -c ssm:configuration-parameter-store-name -s
  ```

  Linux: type the following if you saved the configuration file on the local computer:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
  m onPremise -c file:configuration-file-path -s
  ```

  Windows Server: if you saved the agent configuration file in Systems Manager Parameter Store, use the following command. From the PowerShell console, type the following:

  ```
  ./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m onPremise -c ssm:configuration-
  parameter-store-name -s
  ```

  Windows Server: if you saved the agent configuration file on the local computer, use the following command. From the PowerShell console, type the following:

  ```
  ./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m onPremise -c file:configuration-
  file-path -s
  ```

Verify the Signature of the CloudWatch Agent Package

GPG signature files are included for CloudWatch Agent assets compressed in ZIP archives. The public key is here: https://s3.amazonaws.com/amazoncloudwatch-agent/assets/amazon-cloudwatch-agent.gpg. You can use the public key to verify that the agent's ZIP archive is original and unmodified. First, import the public key with https://gnupg.org/index.html.

To verify the CloudWatch agent package on a Linux server

1. Download the public key.

```
shell$ wget https://s3.amazonaws.com/amazoncloudwatch-agent/assets/amazon-cloudwatch-
agent.gpg
```

2. Import the public key into your keyring.

```
shell$  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

Make a note of the key value, as you need it in the next step. In the preceding example, the key value is 3B789C72.

3. Verify the fingerprint by running the following command, replacing *key-value* with the value from the preceding step:

```
shell$  gpg --fingerprint key-value
pub    2048R/3B789C72 2017-11-14
       Key fingerprint = 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid                    Amazon CloudWatch Agent
```

The fingerprint string should be equal to the following:

```
9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

If the fingerprint string does not match, do not install the agent, and contact Amazon Web Services.

After you have verified the fingerprint, you can use it to verify the signature of the CloudWatch agent package.

4. Download the package signature file using wget. To determine the correct signature file, see CloudWatch Agent Download Links (p. 77).

```
wget https://s3.amazonaws.com/amazoncloudwatch-agent/linux/amd64/latest/
AmazonCloudWatchAgent.sig
```

5. To verify the signature, run `gpg --verify`.

```
shell$ gpg --verify AmazonCloudWatchAgent.sig AmazonCloudWatchAgent.zip
gpg: Signature made Wed 29 Nov 2017 03:00:59 PM PST using RSA key ID 3B789C72
gpg: Good signature from "Amazon CloudWatch Agent"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

If the output includes the phrase `BAD signature`, check whether you performed the procedure correctly. If you continue to get this response, contact Amazon Web Services and avoid unzipping the downloaded agent archive.

Note the warning about trust. A key is only trusted if you or someone that you trust has signed it. This does not mean that the signature is invalid, only that you have not verified the public key.

To verify the CloudWatch agent package on a server running Windows Server

1. Download and install GnuPG for Windows from https://gnupg.org/download/. When installing, include the **Shell Extension (GpgEx)** option.

The remaining steps can be performed in Windows PowerShell.

2. Download the public key.

```
PS> wget https://s3.amazonaws.com/amazoncloudwatch-agent/assets/amazon-cloudwatch-
agent.gpg -OutFile amazon-cloudwatch-agent.gpg
```

3. Import the public key into your keyring.

```
PS>  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

    Make a note of the key value, as you need it in the next step. In the preceding example, the key value is 3B789C72.

4. Verify the fingerprint by running the following command, replacing *key-value* with the value from the preceding step:

```
PS>  gpg --fingerprint key-value
pub   rsa2048 2017-11-14 [SC]
      9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid           [ unknown] Amazon CloudWatch Agent
```

    The fingerprint string should be equal to the following:

```
9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

    If the fingerprint string does not match, do not install the agent, and contact Amazon Web Services.

    After you have verified the fingerprint, you can use it to verify the signature of the CloudWatch agent package.

5. Download the package signature file using wget. To determine the correct signature file, see CloudWatch Agent Download Links (p. 77).

6. To verify the signature, run `gpg --verify`.

```
PS> gpg --verify AmazonCloudWatchAgent.sig AmazonCloudWatchAgent.zip
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

    If the output includes the phrase BAD signature, check whether you performed the procedure correctly. If you continue to get this response, contact Amazon Web Services and avoid unzipping the downloaded agent archive.

    Note the warning about trust. A key is only trusted if you or someone that you trust has signed it. This does not mean that the signature is invalid, only that you have not verified the public key.

Installing CloudWatch Agent on Additional Servers Using Your Agent Configuration

Topics

- Download the CloudWatch Agent on an On-Premises Server (p. 77)
- Modify the Common Configuration and Named Profile for CloudWatch Agent (p. 79)
- Start the CloudWatch Agent (p. 80)

Download the CloudWatch Agent on an On-Premises Server

To download the CloudWatch agent, you can use AWS Systems Manager or the command line.

Download Using Systems Manager

To use Systems Manager Run Command, you must register your on-premises server with Amazon EC2 Systems Manager. For more information, see Setting Up Systems Manager in Hybrid Environments in the *AWS Systems Manager User Guide*.

If you have already registered your server, update your SSM Agent to the latest version.

For information about updating the SSM Agent on a server running Linux, see Install the SSM Agent on Servers and VMs in Your Linux Hybrid Environment in the *AWS Systems Manager User Guide*.

For information about updating the SSM Agent on a server running Windows Server, see Install the SSM Agent on Servers and VMs in Your Windows Hybrid Environment in the *AWS Systems Manager User Guide*.

To use the SSM Agent to download the CloudWatch agent package on an on-premises server

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, select the button next to **AWS-ConfigureAWSPackage**.
5. In the **Targets** area, select the server on which to install the CloudWatch agent. If you do not see a specific server, it might not be configured for Run Command. For more information, see Systems Manager Prerequisites in the *Amazon EC2 User Guide for Windows Instances*.
6. In the **Action** list, choose **Install**.
7. In the **Name** box, type **AmazonCloudWatchAgent**.
8. Leave **Version** blank to install the latest version of the agent.
9. Choose **Run**.

   The agent package is downloaded, and the next steps are to configure and start it.

Download Using an S3 Download Link

To use the command line to download the agent, first choose the download link from this table. The link you choose depends on your architecture and platform.

Arch	Platform	Download Link	Signature File Link
amd64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/ amazoncloudwatch-agent/ amazon_linux/amd64/latest/ amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ amazon_linux/amd64/ latest/amazon-cloudwatch-agent.rpm.sig
amd64	Centos	https://s3.amazonaws.com/ amazoncloudwatch-agent/ centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Redhat	https://s3.amazonaws.com/ amazoncloudwatch-agent/ redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	SUSE	https://s3.amazonaws.com/ amazoncloudwatch-agent/ suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/ amazoncloudwatch-agent/ suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
amd64	Debian	https://s3.amazonaws.com/ amazoncloudwatch-agent/ debian/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/ amazoncloudwatch-agent/ debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Ubuntu	https://s3.amazonaws.com/ amazoncloudwatch-agent/ ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/ amazoncloudwatch-agent/ ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig
amd64	Windows	https://s3.amazonaws.com/ amazoncloudwatch-agent/ windows/amd64/latest/ AmazonCloudWatchAgent.zip	https://s3.amazonaws.com/ amazoncloudwatch-agent/ windows/amd64/latest/ AmazonCloudWatchAgent.zip.sig

To use the command line to download the CloudWatch agent on an on-premises server

1. Make a directory for downloading and unzipping the agent package. For example, `tmp/ AmazonCloudWatchAgent`. Then change into that directory.

2. Download the CloudWatch agent. Use a download link from the previous table. For a Linux server, type the following:

```
wget download-link
```

For a server running Windows Server, download the following file:

```
https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/
AmazonCloudWatchAgent.zip
```

3. After you have downloaded the package, you can optionally use a GPG signature file to verify the package signature. For more information, see Verify the Signature of the CloudWatch Agent Package (p. 74).

4. Unzip the package.

```
unzip AmazonCloudWatchAgent.zip
```

5. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and type:

```
rpm -U ./amazon-cloudwatch-agent.rpm
```

If you downloaded a DEB package on a Linux server, change to the directory containing the package and type:

```
dpkg -i -E ./amazon-cloudwatch-agent.deb
```

On a server running Windows Server, open PowerShell, change to the directory containing the unzipped package, and use the `install.ps1` script to install it.

Modify the Common Configuration and Named Profile for CloudWatch Agent

The CloudWatch agent package you have downloaded includes a configuration file called `common-config.toml`. You can use this file to specify proxy, credential, and region information. On a server running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent` directory.

The default `common-config.toml` is as follows:

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##              Instance role is used for EC2 case by default.
##              AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

All lines are commented out initially. To set the credential profile or proxy settings, remove the # from that line and specify a value. You can edit this file manually, or by using the `RunShellScript` Run Command in Systems Manager:

- **shared_credential_profile** You can specify a name for the named profile that CloudWatch agent is to look for in the AWS credentials and AWS config files. If you don't specify a name here, the CloudWatch agent looks for the default profile name, `AmazonCloudWatchAgent`.
- **shared_credential_file** If you don't want to use the default path, use this line to specify a path to a file containing credentials to use.

- **proxy settings** If your servers use HTTP or HTTPS proxies to contact AWS services, specify those proxies in the `http_proxy` and `https_proxy` fields. If there are URLs that should be excluded from proxying, specify them in the `no_proxy` field, separated by commas.

After modifying `common-config.toml`, you should make sure that the profile name specified, or the default profile name of `AmazonCloudWatchAgent`, exists in the root user's AWS credentials and config files. This profile is used for credential and region information during CloudWatch agent setup. Following is an example of this profile in the credentials file:

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
```

For `my_access_key` and `my_secret_key`, use the keys from the IAM user that does not have the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for the CloudWatch agent, see Create IAM Users to Use with CloudWatch Agent on On-premises Servers (p. 56).

The following is an example of the profile for the configuration file:

```
[AmazonCloudWatchAgent]
region=us-west-1
```

The named profile in the credentials file contains the credentials to be used for the CloudWatch agent. These credentials are used for permissions to write metric data to CloudWatch, and to download information from Systems Manager Parameter Store during the CloudWatch agent installation. You can obtain the credentials to use for this section by creating an IAM user for the CloudWatch agent, as explained previously in this section.

The named profile in the configuration file specifies the region to which the CloudWatch metrics are published, when the CloudWatch agent runs on an on-premises server. When you use the `aws configure` command to modify the profile, do so as the root or administrator.

Following is an example of using the `aws configure` command to create a named profile for the CloudWatch agent. This example assumes that you are using the default profile name of `AmazonCloudWatchAgent`.

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

- Type the following command and follow the prompts:

```
aws configure --profile AmazonCloudWatchAgent
```

Start the CloudWatch Agent

You can start the CloudWatch agent using either Systems Manager Run Command or the command line.

To use the SSM Agent to start the CloudWatch agent on an on-premises server

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.

4. In the **Command document** list, select the button next to **AmazonCloudWatch-ManageAgent**.

5. In the **Targets** area, select the instance where you installed the agent.

6. In the **Action** list, choose **configure**.

7. In the **Mode** list, choose **onPremise**.

8. In the **Optional Configuration Location** box, type the name of the agent configuration file that you created with the wizard and stored in the Parameter Store.

9. Choose **Run**.

   The agent starts with the configuration you specified in the configuration file.

To use the command line to start the CloudWatch agent on an on-premises server

- In this command, `-a fetch-config` causes the agent to load the latest version of the CloudWatch agent configuration file, and `-s` starts the agent.

  Linux: type the following if you saved the configuration file in the Systems Manager Parameter Store:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
  m onPremise -c ssm:configuration-parameter-store-name -s
  ```

  Linux: type the following if you saved the configuration file on the local computer:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
  m onPremise -c file:configuration-file-path -s
  ```

  Windows Server: if you saved the agent configuration file in Systems Manager Parameter Store, use the following command. From the PowerShell console, type the following:

  ```
  ./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m onPremise -c ssm:configuration-
  parameter-store-name -s
  ```

  Windows Server: if you saved the agent configuration file on the local computer, use the following command. From the PowerShell console, type the following:

  ```
  ./amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m onPremise -c file:configuration-
  file-path -s
  ```

Create the CloudWatch Agent Configuration File

Whether you are installing the CloudWatch agent on an Amazon EC2 instance or an on-premises server, you must create the CloudWatch agent configuration file before starting the agent.

The agent configuration file is a JSON file that specifies the metrics and logs that the agent is to collect, including custom metrics. You can create it by using the wizard, or by creating it yourself from scratch. You could also use the wizard to initially create the configuration file, and then modify it manually.

If you create or modify it manually the process is more complex, but you have more control over the metrics collected, and can specify metrics not mentioned in the wizard.

Any time you change the agent configuration file, you must then restart the agent to have the changes take effect.

After you have created a configuration file, you can store it in Systems Manager Parameter Store. This enables you to use the same agent configuration on other servers.

Contents

- Create the CloudWatch Agent Configuration File with the Wizard (p. 82)
- Manually Create or Edit the CloudWatch Agent Configuration File (p. 86)

Create the CloudWatch Agent Configuration File with the Wizard

The agent configuration file wizard, `amazon-cloudwatch-agent-config-wizard`, asks a series of questions, including the following:

- Are you installing the agent on an Amazon EC2 instance or an on-premises server?
- Is the server running Linux or Windows Server?
- Do you want the agent to also send log files to CloudWatch Logs? If so, do you have an existing CloudWatch Logs agent configuration file? If yes, the CloudWatch agent can use this file to determine the logs to collect from the server.
- If you are going to collect metrics from the server, do you want to monitor one of the default sets of metrics, or customize the list of metrics that you collect?
- Do you want to collect custom metrics from your applications or services, using StatsD or collectd?
- Are you migrating from an existing SSM Agent?

The wizard can autodetect the credentials and AWS Region to use, if you have the AWS credentials and configuration files in place before you start the wizard. For more information about these files, see Configuration and Credential Files in the *AWS Systems Manager User Guide*.

The wizard looks for an `AmazonCloudWatchAgent` section such as this in the credentials file:

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_secret_key
aws_secret_access_key=my_access_key
```

If this section exists, the wizard uses these credentials for the CloudWatch agent.

For `my_access_key` and `my_secret_key`, use the keys from the IAM user that has the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for the CloudWatch agent, see Create IAM Users to Use with CloudWatch Agent on On-premises Servers (p. 56).

In the configuration file, you can specify what region the agent sends metrics to, if it is different than the region in the `[default]` section. The default is to publish the metrics to region where the Amazon EC2 instance is located. If the metrics should be published to a different region, specify the region here. In the following example, the metrics are published to the `us-west-1` region.

```
[AmazonCloudWatchAgent]
region=us-west-1
```

CloudWatch Agent Predefined Metric Sets

The wizard is configured with pre-defined sets of metrics, with different detail levels. These sets of metrics are shown in the following tables. For more information about these metrics, see Metrics Collected by the CloudWatch Agent (p. 125).

Amazon EC2 instances running Linux

Detail Level	Metrics Included
Basic	**Mem:** mem_used_percent **Swap:** swap_used_percent
Standard	**CPU:** cpu_usage_idle, cpu_usage_iowait, cpu_usage_user, cpu_usage_system **Disk:** disk_used_percent, disk_inodes_free, diskio_io_time **Mem:** mem_used_percent **Swap:** swap_used_percent
Advanced	**CPU:** cpu_usage_idle, cpu_usage_iowait, cpu_usage_user, cpu_usage_system **Disk:** disk_used_percent, disk_inodes_free **Diskio:** diskio_io_time, diskio_write_bytes, diskio_read_bytes, diskio_writes, diskio_reads **Mem:** mem_used_percent **Netstat:** netstat_tcp_established, netstat_tcp_time_wait **Swap:** swap_used_percent

On-premises servers running Linux

Detail Level	Metrics Included
Basic	**Disk:** disk_used_percent **Diskio:** diskio_write_bytes, diskio_read_bytes, diskio_writes, diskio_reads **Mem:** mem_used_percent **Net:** net_bytes_sent, net_bytes_recv, net_packets_sent, net_packets_recv **Swap:** swap_used_percent
Standard	**CPU:** cpu_usage_idle, cpu_usage_iowait **Disk:** disk_used_percent, disk_inodes_free **Diskio:** diskio_io_time, diskio_write_bytes, diskio_read_bytes, diskio_writes, diskio_reads **Mem:** mem_used_percent **Net:** net_bytes_sent, net_bytes_recv, net_packets_sent, net_packets_recv **Swap:** swap_used_percent
Advanced	**CPU:** cpu_usage_idle, cpu_usage_guest, cpu_usage_iowait, cpu_usage_steal, cpu_usage_user, cpu_usage_system

Detail Level	Metrics Included
	Disk: disk_used_percent, disk_inodes_free
	Diskio: diskio_io_time, diskio_write_bytes, diskio_read_bytes, diskio_writes, diskio_reads
	Mem: mem_used_percent
	Net: net_bytes_sent, net_bytes_recv, net_packets_sent, net_packets_recv
	Netstat: netstat_tcp_established, netstat_tcp_time_wait,
	Swap: swap_used_percent

Amazon EC2 instances running Windows Server

Detail Level	Metrics Included
Basic	**Memory:** Memory % Committed Bytes In Use
	Paging: Paging File % Usage
Standard	**Memory:** Memory % Committed Bytes In Use
	Paging: Paging File % Usage
	Processor: Processor % Idle Time, Processor % Interrupt Time, Processor % User Time,
	PhysicalDisk: PhysicalDisk % Disk Time
	LogicalDisk: LogicalDisk % Free Space
Advanced	**Memory:** Memory % Committed Bytes In Use
	Paging: Paging File % Usage
	Processor: Processor % Idle Time, Processor % Interrupt Time, Processor % User Time
	LogicalDisk: LogicalDisk % Free Space
	PhysicalDisk: PhysicalDisk % Disk Time, PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec
	TCP: TCPv4 Connections Established, TCPv6 Connections Established

On-premises server running Windows Server

Detail Level	Metrics Included
Basic	**Processor:** Processor % Processor Time
	Paging: Paging File % Usage
	LogicalDisk: LogicalDisk % Free Space

Detail Level	Metrics Included
	PhysicalDisk: PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec **Memory:** Memory % Committed Bytes In Use **Network Interface:** Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec, Network Interface Packets Sent/sec, Network Interface Packets Received/sec
Standard	**Paging:** Paging File % Usage **Processor:** Processor_% Processor Time, Processor % Idle Time Processor % Interrupt Time **LogicalDisk:** LogicalDisk % Free Space **PhysicalDisk:** PhysicalDisk % Disk Time, PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec **Memory:** Memory % Committed Bytes In Use **Network Interface:** Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec, Network Interface Packets Sent/sec, Network Interface Packets Received/sec
Advanced	**Paging:**Paging File % Usage **Processor:** Processor % Processor Time, Processor % Idle Time, Processor % Interrupt Time, Processor % User Time **LogicalDisk:** LogicalDisk % Free Space **PhysicalDisk:** PhysicalDisk % Disk Time, PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec **Memory:** Memory % Committed Bytes In Use **Network Interface:** Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec, Network Interface Packets Sent/sec, Network Interface Packets Received/sec **TCP:** TCPv4 Connections Established, TCPv6 Connections Established

Running the CloudWatch Agent Configuration Wizard

To create the CloudWatch agent configuration file

1. Start the CloudWatch agent configuration wizard by typing the following:

    ```
    sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
    ```

    On a server running Windows Server, type the following:

    ```
    cd "C:\Program Files\Amazon\AmazonCloudWatchAgent"
    ```

```
amazon-cloudwatch-agent-config-wizard.exe
```

2. Answer the questions to customize the configuration file for your server.

3. If you are going to use Systems Manager to install and configure the agent, be sure to answer **Yes** when prompted whether to store the file in Systems Manager Parameter Store. You can also choose to store the file in Parameter Store even if you aren't using the SSM Agent to install the CloudWatch agent. To be able to store the file in Parameter Store, you must use an IAM role with sufficient permissions. For more information, see Create IAM Roles and Users for Use With CloudWatch Agent (p. 54).

    If you are storing the configuration file locally, the configuration file `config.json` is stored in the current working directory. You then specify this file location when you start the agent.

Manually Create or Edit the CloudWatch Agent Configuration File

The CloudWatch agent configuration file is a JSON file with three sections: agent, metrics, and logs.

*   The **agent** section includes fields for the overall configuration of the agent. If you use the wizard, it does not create an `agent` section.
*   The **metrics** section specifies the custom metrics for collection and publishing to CloudWatch. If you are using the agent only to collect logs, you can omit the metrics section from the file.
*   The **logs** section specifies what log files are published to CloudWatch Logs. This can include events from the Windows Event Log, if the server runs Windows Server.

The following sections explain the structure and fields of this JSON file. You can also view the schema definition for this configuration file. The schema definition is located at _installation-directory_/`doc/amazon-cloudwatch-agent-schema.json` on Linux servers, and at _installation-directory_/`amazon-cloudwatch-agent-schema.json` on servers running Windows Server.

If you create or edit the JSON file manually, you can give it any name. For simplicity in troubleshooting, we recommend that you name it `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` on a Linux server and `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json` on servers running Windows Server.

CloudWatch Agent Configuration File: Agent Section

The **agent** section can include the fields listed below. The wizard does not create an `agent` section. Instead, the wizard omits it and uses the default values for all fields in this section.

*   **metrics_collection_interval** – Optional. Specifies how often all metrics specified in this configuration file are to be collected. This value can be overridden for specific types of metrics.

    This is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

    If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

    The default is 60.

*   **region** – Specifies the region to use for the CloudWatch endpoint, when an Amazon EC2 instance is being monitored. The metrics collected are sent to this region, such as `us-west-1`. If you omit this field, the agent sends metrics to the region where the Amazon EC2 instance is located.

If you are monitoring an on-premises server, this field is not used, and the agent reads the region from the `awscloudwatchagent` profile of the AWS configuration file.

- **credentials** – Specifies an IAM role to use when sending metrics and logs to a different AWS account. If specified, this field contains one parameter, `role_arn`.

  - **role_arn** – Specifies the ARN of an IAM role to use for authentication when sending metrics and logs to a different AWS account. For more information, see Sending Metrics and Logs to a Different AWS Account (p. 107).

- **debug** – Optional. Specifies running the CloudWatch agent with debug log messages. The default is `false`.

- **logfile** – Specifies the location where the CloudWatch agent writes log messages. If you specify an empty string, the log goes to stderr. If you don't specify this option, the default locations are the following:

  - Linux: `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`

  - Windows Server versions later than Windows Server 2003: `c:\\ProgramData\\Amazon\` `\CloudWatchAgent\\Logs\\amazon-cloudwatch-agent.log`

    Tip
    We suggest you set up log rotation for this file so that it doesn't grow and fill the disk.

The following is an example of an `agent` section:

```
"agent": {
   "metrics_collection_interval": 60,
   "region": "us-west-1",
   "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",
   "debug": false
 }
```

CloudWatch Agent Configuration File: Metrics Section

On servers running either Linux or Windows Server, the **metrics** section includes the following fields:

- **namespace** – Optional. The namespace to use for the metrics collected by the agent. The default is `CWAgent`. The maximum length is 255 characters.

- **append_dimensions** – Optional. Adds Amazon EC2 metric dimensions to all metrics collected by the agent. For each dimension, you must specify a key-value pair, where the key matches an Amazon EC2 dimension: `ImageID:`*image-id*, `InstanceId:`*instance-id*, `InstanceType:`*instance-type*, or `AutoScalingGroupName:`*AutoScaling-group-name*.

  If you specify a value that depends on Amazon EC2 metadata, and you use proxies, you must make sure that the server can access the endpoint for Amazon EC2. For more information about these endpoints, see Amazon Elastic Compute Cloud (Amazon EC2) in the *Amazon Web Services General Reference*.

- **aggregation_dimensions** – Specifies the dimensions on which collected metrics are to be aggregated. For example, if you roll up metrics on the `AutoScalingGroupName` dimension, the metrics from all instances in each Auto Scaling group are aggregated and can be viewed as a whole.

  You can roll up metrics along single or multiple dimensions. For example, specifying `[["InstanceId"], ["InstanceType"], ["InstanceId","InstanceType"]]` aggregates metrics for instance ID singly, instance type singly, and for the combination of the two dimensions.

  You can also specify `[]` to roll up all metrics into one collection, disregarding all dimensions.

- **metrics_collected** – Required. Specifies which metrics are to be collected, including custom metrics collected through StatsD or collectd. This section includes several subsections.

The contents of the `metrics_collected` section depend on whether this configuration file is for a server running Linux or Windows Server.

- **credentials** – Specifies an IAM role to use when sending metrics to a different AWS account. If specified, this field contains one parameter, `role_arn`.

  - **role_arn** – Specifies the ARN of an IAM role to use for authentication when sending metrics to a different AWS account. For more information, see Sending Metrics and Logs to a Different AWS Account (p. 107). If specified here, this overrides the `role_arn` specified in the `agent` section of the configuration file, if any.

Linux

On servers running Linux, the **metrics_collected** section of the configuration file can also contain the following fields:

- **collectd** – Optional. Specifies that you want to retrieve custom metrics using the collectd protocol. You use collectd software to send the metrics to the CloudWatch agent. For more information, see Retrieve Custom Metrics with collectd (p. 104).

- **cpu** – Optional. Specifies that cpu metrics are to be collected. This section is valid only for Linux instances. This section can include as many as three fields:

  - **resources** – Optional. Specifies that per-cpu metrics are to be collected. The only allowed value is `*`. If you include this field and value, per-cpu metrics are collected.

  - **totalcpu** – Optional. Specifies whether to report cpu metrics aggregated across all cpu cores. The default is true.

    If you use the wizard to create your configuration file, the wizard sets this option to false.

  - **measurement** – Specifies the array of cpu metrics to be collected. Possible values are `time_active`, `time_guest`, `time_guest_nice`, `time_idle`, `time_iowait`, `time_irq`, `time_nice`, `time_softirq`, `time_steal`, `time_system`, `time_user`, `usage_active`, `usage_guest`, `usage_guest_nice`, `usage_idle`, `usage_iowait`, `usage_irq`, `usage_nice`, `usage_softirq`, `usage_steal`, `usage_system`, and `usage_user`. This field is required if you include `cpu`.

    By default, the unit for `cpu_usage_*` metrics is `Percent`, and `cpu_time_*` metrics do not have a unit.

    Within the entry for each individual metric, you may optionally specify one or both of the following:

    - **rename** – Specifies a different name for this metric.

    - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

  - **metrics_collection_interval** – Optional. Specifies how often to collect the cpu metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

    If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

  - **append_dimensions** – Optional. Additional dimensions to use for only the cpu metrics. If you specify this field, it is used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics collected by the agent.

- **disk** – Optional. Specifies that disk metrics are to be collected. This section is valid only for Linux instances. This section can include as many as two fields:

- **resources** – Optional. Specifies an array of disk mount points. This field limits CloudWatch to collect metrics from only the listed mount points. You can specify * as the value to collect metrics from all mount points. The default is to collect metrics from all mount points.

- **measurement** – Specifies the array of disk metrics to be collected. Possible values are `free`, `total`, `used`, `used_percent`, `inodes_free`, `inodes_used`, and `inodes_total`. This field is required if you include `disk`.

  To see the default units for each `disk` metric, see Metrics Collected by the CloudWatch Agent on Linux Instances (p. 126).

  Within the entry for each individual metric, you may optionally specify one or both of the following:

  - **rename** – Specifies a different name for this metric.

  - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

- **ignore_file_system_types** – Specifies file system types to exclude when collecting disk metrics. Valid values include `sysfs`, `devtmpfs`, and so on.

- **metrics_collection_interval** – Optional. Specifies how often to collect the disk metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

  This is specified in seconds.

  If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information, see High-Resolution Metrics (p. 45).

- **append_dimensions** – Optional. Additional dimensions to use for only the disk metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.

- **diskio** – Optional. Specifies that diskio metrics are to be collected. This section is valid only for Linux instances. This section can include as many as two fields:

  - **resources** – Optional. If you specify an array of devices, CloudWatch collects metrics from only those devices. Otherwise, metrics for all devices are collected. You can also specify * as the value to collect metrics from all devices.

  - **measurement** – Specifies the array of diskio metrics to be collected. Possible values are `reads`, `writes`, `read_bytes`, `write_bytes`, `read_time`, `write_time`, `io_time`, and `iops_in_progress`. This field is required if you include `diskio`.

    To see the default units for each `diskio` metric, see Amazon EC2 Metrics and Dimensions (p. 152).

    Within the entry for each individual metric, you may optionally specify one or both of the following:

    - **rename** – Specifies a different name for this metric.

    - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

  - **metrics_collection_interval** – Optional. Specifies how often to collect the diskio metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This is specified in seconds.

    If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

  - **append_dimensions** – Optional. Additional dimensions to use for only the diskio metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.

- **swap** – Optional. Specifies that swap memory metrics are to be collected. This section is valid only for Linux instances. This section can include one field:

  - **measurement** – Specifies the array of swap metrics to be collected. Possible values are `free`, `used`, and `used_percent`. This field is required if you include `swap`.

    To see the default units for each `swap` metric, see Metrics Collected by the CloudWatch Agent on Linux Instances (p. 126).

    Within the entry for each individual metric, you may optionally specify one or both of the following:

    - **rename** Specifies a different name for this metric.

    - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

  - **metrics_collection_interval** – Optional. Specifies how often to collect the swap metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This is specified in seconds.

    If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

  - **append_dimensions** Optional. Additional dimensions to use for only the swap metrics. If you specify this field, it is used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics collected by the agent. It is collected as a high-resolution metric.

- **mem** – Optional. Specifies that memory metrics are to be collected. This section is valid only for Linux instances. This section can include one field:

  - **measurement** – Specifies the array of swap metrics to be collected. Possible values are `active`, `available`, `available_percent`, `buffered`, `cached`, `free`, `inactive`, `total`, `used`, and `used_percent`. This field is required if you include `mem`.

    To see the default units for each `mem` metric, see Metrics Collected by the CloudWatch Agent on Linux Instances (p. 126).

    Within the entry for each individual metric, you may optionally specify one or both of the following:

    - **rename** – Specifies a different name for this metric.

    - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

  - **metrics_collection_interval** – Optional. Specifies how often to collect the mem metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This is specified in seconds.

    If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

  - **append_dimensions** – Optional. Additional dimensions to use for only the mem metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.

- **net** – Optional. Specifies that networking metrics are to be collected. This section is valid only for Linux instances. This section can include as many as two fields:

  - **resources** – Optional. If you specify an array of network interfaces, CloudWatch collects metrics from only those interfaces. Otherwise, metrics for all devices are collected. You can also specify * as the value to collect metrics from all interfaces.

- **measurement** – Specifies the array of networking metrics to be collected. Possible values are `bytes_sent`, `bytes_recv`, `drop_in`, `drop_out`, `err_in`, `err_out`, `packets_sent`, and `packets_recv`. This field is required if you include `net`.

  To see the default units for each `net` metric, see Metrics Collected by the CloudWatch Agent on Linux Instances (p. 126).

  Within the entry for each individual metric, you may optionally specify one or both of the following:

  - **rename** – Specifies a different name for this metric.

  - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

- **metrics_collection_interval** – Optional. Specifies how often to collect the net metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

  This is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

  If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

- **append_dimensions** – Optional. Additional dimensions to use for only the net metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.

- **netstat** – Optional. Specifies that TCP connection state and UDP connection metrics are to be collected. This section is valid only for Linux instances. This section can include one field:

  - **measurement** – Specifies the array of netstat metrics to be collected. Possible values are `tcp_close`, `tcp_close_wait`, `tcp_closing`, `tcp_established`, `tcp_fin_wait1`, `tcp_fin_wait2`, `tcp_last_ack`, `tcp_listen`, `tcp_none`, `tcp_syn_sent`, `tcp_syn_recv`, `tcp_time_wait`, and `udp_socket`. This field is required if you include `netstat`.

    To see the default units for each `netstat` metric, see Metrics Collected by the CloudWatch Agent on Linux Instances (p. 126).

    Within the entry for each individual metric, you may optionally specify one or both of the following:

    - **rename** – Specifies a different name for this metric.

    - **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

  - **metrics_collection_interval** – Optional. Specifies how often to collect the netstat metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This is specified in seconds.

    If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see High-Resolution Metrics (p. 45).

  - **append_dimensions** – Optional. Additional dimensions to use for only the netstat metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.

- **processes** – Optional. Specifies that process metrics are to be collected. This section is valid only for Linux instances. This section can include one field:

  - **measurement** – Specifies the array of processes metrics to be collected. Possible values are `blocked`, `dead`, `idle`, `paging`, `running`, `sleeping`, `stopped`, `total`, `total_threads`, `wait`, and `zombies`. This field is required if you include `processes`.

For all `processes` metrics, the default unit is `Count`.

Within the entry for each individual metric, you may optionally specify one or both of the following:

- **rename** – Specifies a different name for this metric.
- **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum]().

- **metrics_collection_interval** – Optional. Specifies how often to collect the processes metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

  This is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

  If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information, see High-Resolution Metrics (p. 45).

- **append_dimensions** – Optional. Additional dimensions to use for only the process metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.

- **statsd** – Optional. Specifies that you want to retrieve custom metrics using the StatsD protocol. The CloudWatch agent acts as a daemon for the protocol. You use any standard StatsD client to send the metrics to the CloudWatch agent. For more information, see Retrieve Custom Metrics with StatsD (p. 103).

The following is an example of a `metrics` section for a Linux server. In this example, three CPU metrics, three netstat metrics, and three process metrics are collected, and the agent is set up to receive additional metrics from a collectd client.

```
"metrics": {
  "metrics_collected": {
    "collectd": {},
    "cpu": {
      "resources": [
        "*"
      ],
      "measurement": [
        {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent"},
        {"name": "cpu_usage_nice", "unit": "Percent"},
        "cpu_usage_guest"
      ],
      "totalcpu": false,
      "metrics_collection_interval": 10,
      "append_dimensions": {
        "test": "test1",
        "date": "2017-10-01"
      }
    },
    "netstat": {
      "measurement": [
        "tcp_established",
        "tcp_syn_sent",
        "tcp_close"
      ],
      "metrics_collection_interval": 60
    },
    "processes": {
      "measurement": [
        "running",
        "sleeping",
```

```
          "dead"
        ]
      }
    },
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    },
    "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId", "InstanceType"],
[]]
  }
```

Windows Server

In the **metrics_collected** section for Windows Server, you can have subsections for each Windows performance object, such as `Memory`, `Processor`, and `LogicalDisk`. For information about what objects and counters are available, see the Microsoft Windows documentation.

Within the subsection for each object, you specify a `measurement` array of the counters to collect. The `measurement` array is required for each object that you specify in the configuration file. You can also specify a `resources` field to name the instances from which to collect metrics. You can also specify `*` for `resources`, to collect separate metrics for every instance. If you omit `resources`, the data for all instances is aggregated into one set.

Within each object section, you can also specify the following optional fields:

- **metrics_collection_interval** – Optional. Specifies how often to collect the metrics for this object, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

  This is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

  If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information, see High-Resolution Metrics (p. 45).
- **append_dimensions** –Optional. Additional dimensions to use for only the metrics for this object. If you specify this field, it is used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics collected by the agent.

Within each counter section, you can also specify the following optional fields:

- **rename** – Specifies a different name to be used in CloudWatch for this metric.
- **unit** – Specifies the unit to use for this metric. The unit you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

To retrieve custom metrics using StatsD, you include a **StatsD** subsection in **metrics_collected**. The CloudWatch agent acts as a daemon for the protocol. You use any standard StatsD client to send the metrics to the CloudWatch agent.

The following is an example `metrics` section for use on Windows Server. In this example, many Windows metrics are collected, and the computer is also set to receive additional metrics from a StatsD client.

```
"metrics": {
    "metrics_collected": {
        "statsd": {},
```

```
        "Processor": {
          "measurement": [
            {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
            "% Interrupt Time",
            "% User Time",
            "% Processor Time"
          ],
          "resources": [
            "*"
          ],
          "append_dimensions": {
            "d1": "win_foo",
            "d2": "win_bar"
          }
        },
        "LogicalDisk": {
          "measurement": [
            {"name": "% Idle Time", "unit": "Percent"},
            {"name": "% Disk Read Time", "rename": "DISK_READ"},
            "% Disk Write Time"
          ],
          "resources": [
            "*"
          ]
        },
        "Memory": {
          "metrics_collection_interval": 5,
          "measurement": [
            "Available Bytes",
            "Cache Faults/sec",
            "Page Faults/sec",
            "Pages/sec"
          ],
          "append_dimensions": {
            "d3": "win_bo"
          }
        },
        "Network Interface": {
          "metrics_collection_interval": 5,
          "measurement": [
            "Bytes Received/sec",
            "Bytes Sent/sec",
            "Packets Received/sec",
            "Packets Sent/sec"
          ],
          "resources": [
            "*"
          ],
          "append_dimensions": {
            "d3": "win_bo"
          }
        },
        "System": {
          "measurement": [
            "Context Switches/sec",
            "System Calls/sec",
            "Processor Queue Length"
          ],
          "append_dimensions": {
            "d1": "win_foo",
            "d2": "win_bar"
          }
        }
      },
      "append_dimensions": {
        "ImageId": "${aws:ImageId}",
```

```
        "InstanceId": "${aws:InstanceId}",
        "InstanceType": "${aws:InstanceType}",
        "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    },
    "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],[]]
    }
}
```

CloudWatch Agent Configuration File: Logs Section

The **logs** section includes the following fields:

- **logs_collected** – Required if the **logs** section is included. Specifies which log files and Windows event logs are to be collected from the server. It can include two fields, **files** and **windows_events**.

  - **files** specifies which regular log files are to be collected by the CloudWatch agent. It contains one field, **collect_list**, which further defines these files.

    - **collect_list** – Required if **files** is included. Contains an array of entries, each of which specifies one log file to collect. Each of these entries can include the following fields:

      - **file_path** – Specifies the path of the log file to upload to CloudWatch Logs. Standard Unix glob matching rules are accepted, with the addition of `**` as a *super asterisk*. For example, specifying `/var/log/**.log` causes all `.log` files in the `/var/log` directory tree to be collected. For more examples, see Glob Library.

        The standard asterisk can also be used as a standard wildcard. For example, `/var/log/system.log*` matches files such as `system.log_1111`, `system.log_2222`, and so on in `/var/log`.

        Only the latest file is pushed to CloudWatch Logs based on file modification time. We recommend that you use wildcards to specify a series of files of the same type, such as `access_log.2018-06-01-01` and `access_log.2018-06-01-02`, but not multiple kinds of files, such as `access_log_80` and `access_log_443`. To specify multiple kinds of files, add another log stream entry to the agent configuration file so each kind of log file goes to a different log stream.

      - **log_group_name** – Optional. Specifies what to use as the log group name in CloudWatch Logs. Allowed characters include a-z, A-Z, 0-9, '_' (underscore), '-' (hyphen), '/' (forward slash), and '.' (period).

        We recommend that you specify this field to prevent confusion. If you omit this field, the file path up to the final dot is used as the log group name. For example, if the file path is `/tmp/TestLogFile.log.2017-07-11-14`, the log group name is `/tmp/TestLogFile.log`.

      - **log_stream_name** – Optional. Specifies what to use as the log stream name in CloudWatch Logs. As part of the name, you can use `{instance_id}`, `{hostname}`, `{local_hostname}`, and `{ip_address}` as variables within the name. `{hostname}` retrieves the hostname from the EC2 metadata, while `{local_hostname}` uses the hostname from the network configuration file.

        If you omit this field, the default of `{instance_id}` is used. A log stream is created automatically if it does not already exist.

      - **timezone** – Optional. Specifies the time zone to use when putting time stamps on log events. The valid values are `UTC` and `Local`. The default is `Local`.

      - **timestamp_format** – Optional. Specifies the time stamp format, using plaintext and special symbols that start with %. If you omit this field, the current time is used. If you use this field, you can use the following as part of the format:

        `%y`

          Year without century as a zero-padded decimal number

`%Y`

Year with century as a decimal number

`%b`

Month as the locale's abbreviated name

`%B`

Month as the locale's full name

`%m`

Month as a zero-padded decimal number

`%-m`

Month as a decimal number (not zero-padded)

`%d`

Day of the month as a zero-padded decimal number

`%-d`

Day of the month as a decimal number (not zero-padded)

`%A`

Full name of weekday, such as `Monday`

`%a`

Abbreviation of weekday, such as `Mon`

`%H`

Hour (in a 24-hour clock) as a zero-padded decimal number

`%I`

Hour (in a 12-hour clock) as a zero-padded decimal number

`%-I`

Hour (in a 12-hour clock) as a decimal number (not zero-padded)

`%p`

AM or PM

`%M`

Minutes as a zero-padded decimal number

`%-M`

Minutes as a decimal number (not zero-padded)

`%S`

Seconds as a zero-padded decimal number

`%-S`

Seconds as a decimal number (not zero padded)

`%Z`

Time zone, for example `PST`

`%z`

Time zone, expressed as the offset between the local time zone and UTC. For example, `-0700`. Only this format is supported. For example, `-07:00` is not a valid format.

- **multi_line_start_pattern** – Specifies the pattern for identifying the start of a log message. A log message is made of a line that matches the pattern and any following lines that don't match the pattern.

  If you omit this field, multi-line mode is disabled, and any line that begins with a non-whitespace character closes the previous log message and starts a new log message.

  If you include this field, you can specify `{timestamp_format}` to use the same regular expression as your time stamp format. Otherwise, you can specify a different regular expression for CloudWatch Logs to use to determine the start lines of multi-line entries.

- **encoding** – Specified the encoding of the log file so that it can be read correctly. If you specify an incorrect coding, there might be data loss because characters that cannot be decoded are replaced with other characters.

  The default is utf-8. Below are all possible values:

  ```
  ascii, big5, euc-jp, euc-kr, gbk, gb18030, ibm866, iso2022-jp,
  iso8859-2, iso8859-3, iso8859-4, iso8859-5, iso8859-6, iso8859-7,
  iso8859-8, iso8859-8-i, iso8859-10, iso8859-13, iso8859-14, iso8859-15,
  iso8859-16, koi8-r, koi8-u, macintosh, shift_jis, utf-8, utf-16,
  windows-874, windows-1250, windows-1251, windows-1252, windows-1253,
  windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, x-
  mac-cyrillic
  ```

- The **windows_events** section specifies the type of Windows events to collect from servers running Windows Server. It includes the following fields:

  - **collect_list** – Required if **windows_events** is included. Specifies the types and levels of Windows events to be collected. Each log to be collected has an entry in this section, which can include the following fields:

    - **event_name** – Specifies the type of Windows events to log. For example, `System`, `Security`, `Application`, and so on. This field is required for each type of Windows event to log.

    - **event_levels** – Specifies the levels of event to log. You must specify each level to log. Possible values include `INFORMATION`, `WARNING`, `ERROR`, `CRITICAL`, and `VERBOSE`. This field is required for each type of Windows event to log.

    - **log_group_name** – Required. Specifies what to use as the log group name in CloudWatch Logs.

    - **log_stream_name** – Optional. Specifies what to use as the log stream name in CloudWatch Logs. As part of the name, you can use `{instance_id}`, `{hostname}`, `{local_hostname}`, and `{ip_address}` as variables within the name. `{hostname}` retrieves the hostname from the EC2 metadata, while `{local_hostname}` uses the hostname from the network configuration file.

      If you omit this field, the default of `{instance_id}` is used. If a log stream does not already exist, it is created automatically .

    - **event_format** – Optional. Specifies the format to use when storing Windows events in CloudWatch Logs. `xml` uses the XML format as in Windows Event Viewer. `text` uses the legacy CloudWatch Logs agent format.

- **log_stream_name** – Required. Specifies the default log stream name to be used for any logs or Windows events that do not have individual log stream names defined in their entry in **collect_list**.

- **credentials** – Specifies an IAM role to use when sending logs to a different AWS account. If specified, this field contains one parameter, `role_arn`.

  - **role_arn** – Specifies the ARN of an IAM role to use for authentication when sending logs to a different AWS account. For more information, see Sending Metrics and Logs to a Different AWS

Account (p. 107). If specified here, this overrides the `role_arn` specified in the `agent` section of the configuration file, if any.

The following is an example of a `logs` section:

```
"logs":{
   "logs_collected":{
      "files":{
         "collect_list":[
            {
               "file_path":"c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log",
               "log_group_name":"amazon-cloudwatch-agent.log",
               "log_stream_name":"my_log_stream_name_1",
               "timestamp_format":"%H:%M:%S %y %b %-d"
            },
            {
               "file_path":"c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\test.log",
               "log_group_name":"test.log",
               "log_stream_name":"my_log_stream_name_2"
            }
         ]
      },
      "windows_events":{
         "collect_list":[
            {
               "event_name":"System",
               "event_levels":[
                  "INFORMATION",
                  "ERROR"
               ],
               "log_group_name":"System",
               "log_stream_name":"System"
            },
            {
               "event_name":"Application",
               "event_levels":[
                  "INFORMATION",
                  "ERROR"
               ],
               "log_group_name":"Application",
               "log_stream_name":"Application"
            }
         ]
      }
   },
   "log_stream_name":"my_log_stream_name"
}
```

CloudWatch Agent Configuration File: Complete Examples

The following is an example of a complete agent configuration file for a Linux server.

```
{
   "agent": {
      "metrics_collection_interval": 10,
      "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
   },
   "metrics": {
      "metrics_collected": {
         "cpu": {
```

```
          "resources": [
            "*"
          ],
          "measurement": [
            {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent"},
            {"name": "cpu_usage_nice", "unit": "Percent"},
            "cpu_usage_guest"
          ],
          "totalcpu": false,
          "metrics_collection_interval": 10,
          "append_dimensions": {
            "customized_dimension_key_1": "customized_dimension_value_1",
            "customized_dimension_key_2": "customized_dimension_value_2"
          }
        },
        "disk": {
          "resources": [
            "/",
            "/tmp"
          ],
          "measurement": [
            {"name": "free", "rename": "DISK_FREE", "unit": "Gigabytes"},
            "total",
            "used"
          ],
           "ignore_file_system_types": [
            "sysfs", "devtmpfs"
          ],
          "metrics_collection_interval": 60,
          "append_dimensions": {
            "customized_dimension_key_3": "customized_dimension_value_3",
            "customized_dimension_key_4": "customized_dimension_value_4"
          }
        },
        "diskio": {
          "resources": [
            "*"
          ],
          "measurement": [
            "reads",
            "writes",
            "read_time",
            "write_time",
            "io_time"
          ],
          "metrics_collection_interval": 60
        },
        "swap": {
          "measurement": [
            "swap_used",
            "swap_free",
            "swap_used_percent"
          ]
        },
        "mem": {
          "measurement": [
            "mem_used",
            "mem_cached",
            "mem_total"
          ],
          "metrics_collection_interval": 1
        },
        "net": {
          "resources": [
            "eth0"
          ],
```

```
              "measurement": [
                "bytes_sent",
                "bytes_recv",
                "drop_in",
                "drop_out"
              ]
            },
            "netstat": {
              "measurement": [
                "tcp_established",
                "tcp_syn_sent",
                "tcp_close"
              ],
              "metrics_collection_interval": 60
            },
            "processes": {
              "measurement": [
                "running",
                "sleeping",
                "dead"
              ]
            }
          },
          "append_dimensions": {
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}",
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
          },
          "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],[]]
        },
        "logs": {
          "logs_collected": {
            "files": {
              "collect_list": [
                {
                  "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
                  "log_group_name": "amazon-cloudwatch-agent.log",
                  "log_stream_name": "amazon-cloudwatch-agent.log",
                  "timezone": "UTC"
                },
                {
                  "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
                  "log_group_name": "test.log",
                  "log_stream_name": "test.log",
                  "timezone": "Local"
                }
              ]
            }
          },
          "log_stream_name": "my_log_stream_name"
        }
      }
```

The following is an example of a complete agent configuration file for a server running Windows Server.

```
{
      "agent": {
        "metrics_collection_interval": 60,
        "logfile": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log"
      },
      "metrics": {
        "metrics_collected": {
```

```
        "Processor": {
          "measurement": [
            {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
            "% Interrupt Time",
            "% User Time",
            "% Processor Time"
          ],
          "resources": [
            "*"
          ],
          "append_dimensions": {
            "customized_dimension_key_1": "customized_dimension_value_1",
            "customized_dimension_key_2": "customized_dimension_value_2"
          }
        },
        "LogicalDisk": {
          "measurement": [
            {"name": "% Idle Time", "unit": "Percent"},
            {"name": "% Disk Read Time", "rename": "DISK_READ"},
            "% Disk Write Time"
          ],
          "resources": [
            "*"
          ]
        },
        "customizedObjectName": {
          "metrics_collection_interval": 60,
          "customizedCounterName": [
            "metric1",
            "metric2"
          ],
          "resources": [
            "customizedInstaces"
          ]
        },
        "Memory": {
          "metrics_collection_interval": 5,
          "measurement": [
            "Available Bytes",
            "Cache Faults/sec",
            "Page Faults/sec",
            "Pages/sec"
          ]
        },
        "Network Interface": {
          "metrics_collection_interval": 5,
          "measurement": [
            "Bytes Received/sec",
            "Bytes Sent/sec",
            "Packets Received/sec",
            "Packets Sent/sec"
          ],
          "resources": [
            "*"
          ],
          "append_dimensions": {
            "customized_dimension_key_3": "customized_dimension_value_3"
          }
        },
        "System": {
          "measurement": [
            "Context Switches/sec",
            "System Calls/sec",
            "Processor Queue Length"
          ]
        }
```

```
        },
        "append_dimensions": {
          "ImageId": "${aws:ImageId}",
          "InstanceId": "${aws:InstanceId}",
          "InstanceType": "${aws:InstanceType}",
          "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
        },
        "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],[]]
      },
      "logs": {
        "logs_collected": {
          "files": {
            "collect_list": [
              {
                "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log",
                "log_group_name": "amazon-cloudwatch-agent.log",
                "timezone": "UTC"
              },
              {
                "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\test.log",
                "log_group_name": "test.log",
                "timezone": "Local"
              }
            ]
          },
          "windows_events": {
            "collect_list": [
              {
                "event_name": "System",
                "event_levels": [
                  "INFORMATION",
                  "ERROR"
                ],
                "log_group_name": "System",
                "log_stream_name": "System",
                "event_format": "xml"
              },
              {
                "event_name": "Application",
                "event_levels": [
                  "WARNING",
                  "ERROR"
                ],
                "log_group_name": "Application",
                "log_stream_name": "Application",
                "event_format": "xml"
              }
            ]
          }
        },
        "log_stream_name": "example_log_stream_name"
      }
    }
```

Upload the CloudWatch Agent Configuration File to Systems Manager Parameter Store

If you are installing the CloudWatch agent on an Amazon EC2 instance or on an on-premises server that has the SSM Agent installed, then after you manually edit the CloudWatch agent configuration file you must upload it to Systems Manager Parameter Store. To do so, you use the Systems Manager `put-parameter` command.

To be able to store the file in Parameter Store, you must use an IAM role with sufficient permissions. For more information, see Create IAM Roles and Users for Use With CloudWatch Agent (p. 54).

Use the following command, where `parameter name` is the name to be used for this file in Parameter Store, and `configuration_file_pathname` is the path and filename of the configuration file you have edited.

```
aws ssm put-parameter --name "parameter name" --type "String" --value
 file://configuration_file_pathname
```

Retrieve Custom Metrics with StatsD

You can retrieve custom metrics from your applications or services using the CloudWatch agent with the StatsD protocol. StatsD is supported on both Linux servers and servers running Windows Server. CloudWatch supports the following StatsD format:

```
MetricName:value|type|@sample_rate|#tag1:
  value,tag1...
```

- `MetricName` – A string with no colons, bars, # characters, or @ characters.
- `value` – This can be either integer or float.
- `type` – Specify `c` for counter, `g` for gauge, `ms` for timer, `h` for histogram, or `s` for set.
- `sample_rate` – (Optional) A float between 0 and 1, inclusive. Use only for counter, histogram, and timer metrics. The default is 1 (sampling 100% of the time).
- `tags` – (Optional) A comma-separated list of tags. StatsD tags are similar to dimensions in CloudWatch. Use colons for key/value tags, such as `env:prod`.

You can use any StatsD client that follows this format to send the metrics to the CloudWatch agent. For more information about some of the available StatsD clients, see the StatsD client page on GitHub.

The basic configuration for collecting these custom metrics with the CloudWatch agent is to add a **"statsd": {}** line to the **metrics_collected** section of the agent configuration file. You can add this line manually. If you use the wizard to create the configuration file, it is done for you. For more information, see Create the CloudWatch Agent Configuration File (p. 81).

The StatsD default configuration works for most users. There are three optional fields you can add to the **statsd** section of the agent configuration file as needed:

- **service_address:** The service address to which the CloudWatch agent should listen. The format is `ip:port`. If you omit the IP address, the local host `127.0.0.1` is used for the IP address. Only the UDP format is supported, so you do not need to specify a UDP prefix.

  The default is `:8125`, which translates to the 8125 port of the local host.
- **metrics_collection_interval:** How often in seconds that the StatsD plugin runs and collects metrics. The default is 10 seconds. The range is 1 to 172,000.
- **metrics_aggregation_interval:** How often in seconds CloudWatch aggregates metrics into single data points. The default is 60 seconds.

  For example, if `metrics_collection_interval` is 10 and `metrics_aggregation_interval` is 60, CloudWatch collects data every 10 seconds. After each minute, the six data readings from that minute are aggregated into a single data point, which is sent to CloudWatch.

  The range is 0 to 172,000. Setting `metrics_aggregation_interval` to 0 disables the aggregation of StatsD metrics.

The following is an example of the **statsd** section of the agent configuration file, using the default port and custom collection and aggregation intervals.

```
{
    "metrics":{
        "metrics_collected":{
            "statsd":{
                "service_address":":8125",
                "metrics_collection_interval":60,
                "metrics_aggregation_interval":300
            }
        }
    }
}
```

Retrieve Custom Metrics with collectd

You can retrieve custom metrics from your applications or services using the CloudWatch agent with the collectd protocol. collectd is supported only on Linux servers. You use collectd software to send the metrics to the CloudWatch agent.

collectd software is not installed automatically on every server. For more information about collectd and downloading collectd software, see the Download page for collectd.

Basic configuration for collecting these custom metrics with the CloudWatch agent is to add a **"collectd": {}** line to the **metrics_collected** section of the agent configuration file. You can add this line manually. If you use the wizard to create the configuration file, it is done for you. For more information, see Create the CloudWatch Agent Configuration File (p. 81).

Optional parameters are also available. If you are using collectd and you do not use `/etc/collectd/auth_file` as your **collectd_auth_file**, you must set some of these options.

- **service_address:** The service address to which the CloudWatch agent should listen. The format is `"udp://ip:port`. The default is `udp://127.0.0.1:25826`
- **name_prefix:** A prefix to attach to the beginning of the name of each collectd metric. The default is `collectd_`. The maximum length is 255 characters.
- **collectd_security_level:** Sets the security level for network communication. The default is **Encrypt**.

  Encrypt specifies that only encrypted data is accepted. **Sign** specifies that only signed and encrypted data is accepted. **None** specifies that all data is accepted. If you specify a value for **collectd_auth_file**, encrypted data is decrypted if possible.

  For more information, see Client setup and Possible interactions in the collectd Wiki.
- **collectd_auth_file** Sets a file in which user names are mapped to passwords. These passwords are used to verify signatures and to decrypt encrypted network packets. If given, signed data is verified and encrypted packets are decrypted. Otherwise, signed data is accepted without checking the signature and encrypted data cannot be decrypted.

  The default is `/etc/collectd/auth_file`

  If **collectd_security_level** is set to **None**, this is optional. If you set **collectd_security_level** to `Encrypt` or **Sign**, you must specify **collectd_auth_file**.

  For the format of the auth file, each line is a user name followed by a colon and any number of spaces followed by the password. For example:

  `user1: user1_password`

```
user2: user2_password
```

- **collectd_types_db:** A list of one or more files that contain the dataset descriptions. The list must be surrounded by brackets, even if there is just one entry in the list. Each entry in the list must be surrounded by double quotes. If there are multiple entries, separate them with commas. The default is `["/usr/share/collectd/types.db"]`. For more information, see https://collectd.org/documentation/manpages/types.db.5.shtml.
- **metrics_aggregation_interval:** How often in seconds CloudWatch aggregates metrics into single data points. The default is 60 seconds. The range is 0 to 172,000. Setting it to 0 disables the aggregation of collectd metrics.

The following is an example of the collectd section of the agent configuration file.

```
{
    "metrics":{
        "metrics_collected":{
            "collectd":{
                "name_prefix":"My_collectd_metrics_",
                "metrics_aggregation_interval":120
            }
        }
    }
}
```

Common Scenarios with CloudWatch Agent

The following sections outline how to complete some common configuration and customization tasks when using the CloudWatch agent.

Topics

Adding Custom Dimensions to Metrics Collected by the CloudWatch Agent

To add custom dimensions such as tags to metrics collected by the agent, add the `append_dimensions` field to the section of the agent configuration file that lists those metrics.

For example, the following example section of the configuration file adds a custom dimension named `stackName` with a value of `Prod` to the `cpu` and `disk` metrics collected by the agent.

```
"cpu":{
  "resources":[
    "*"
  ],
  "measurement":[
    "cpu_usage_guest",
    "cpu_usage_nice",
    "cpu_usage_idle"
```

```
    ],
    "totalcpu":false,
    "append_dimensions":{
      "stackName":"Prod"
    }
  },
  "disk":{
    "resources":[
      "/",
      "/tmp"
    ],
    "measurement":[
      "total",
      "used"
    ],
    "append_dimensions":{
      "stackName":"Prod"
    }
  }
```

Remember that any time you change the agent configuration file, you must then restart the agent to have the changes take effect.

Aggregating or Rolling Up Metrics Collected by the CloudWatch Agent

To aggregate or "roll up" metrics collected by the agent, add an `aggregation_dimensions` field to the section for that metric in the agent configuration file.

For example, the following configuration file snippet rolls up metrics on the `AutoScalingGroupName` dimension. The metrics from all instances in each Auto Scaling group are aggregated and can be viewed as a whole.

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"]]
}
```

To roll up along the combination of each `InstanceId` and `InstanceType` dimensions in addition to rolling up on the Auto Scaling group name, add the following:

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId", "InstanceType"]]
}
```

To roll up metrics into one collection instead, use `[ ]`.

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [[]]
}
```

Remember that any time you change the agent configuration file, you must then restart the agent to have the changes take effect.

Collecting High-Resolution Metrics With the CloudWatch agent

The `metrics_collection_interval` field specifies the time interval for the metrics collected, in seconds. By specifying a value of less than 60 for this field, the metrics are collected as high-resolution metrics.

For example, if your metrics should all be high-resolution and collected every 10 seconds, specify 10 as the value for `metrics_collection_interval` under the `agent` section as a global metrics collection interval:

```
"agent": {
  "metrics_collection_interval": 10
}
```

Alternatively, the following example sets the `cpu` metrics to be collected every second, while all other metrics are collected every minute.

```
"agent":{
  "metrics_collection_interval": 60
},
"metrics":{
  "metrics_collected":{
    "cpu":{
      "resources":[
        "*"
      ],
      "measurement":[
        "cpu_usage_guest"
      ],
      "totalcpu":false,
      "metrics_collection_interval": 1
    },
    "disk":{
      "resources":[
        "/",
        "/tmp"
      ],
      "measurement":[
        "total",
        "used"
      ]
    }
  }
}
```

Remember that any time you change the agent configuration file, you must then restart the agent to have the changes take effect.

Sending Metrics and Logs to a Different AWS Account

To have the CloudWatch agent send the metrics, logs, or both to a different AWS account, specify a `role_arn` parameter in the agent configuration file on the sending server. The `role_arn` value specifies an IAM role in the sending account that the agent uses when sending data to the target account. This role enables the sending account to assume a corresponding role in the target account when delivering the metrics or logs to the target account.

You can also specify two separate `role_arn` strings in the agent configuration file: one to use when sending metrics, and another for sending logs.

The following example of part of the `agent` section of the configuration file sets the agent to use
`CrossAccountAgentRole` when sending metrics and logs to a different AWS account.

```
{
  "agent": {
    "credentials": {
      "role_arn": "CrossAccountAgentRole"
    }
  },
  .....
}
```

Alternatively, the following example sets different roles for the sending account to use for sending
metrics and logs:

```
"metrics": {
    "credentials": {
        "role_arn": "RoleToSendMetrics"
    },
    "metrics_collected": {....
```

```
"logs": {
    "credentials": {
        "role_arn": "RoleToSendLogs"
    },
    ....
```

Policies Needed

When you specify a `role_arn` in the agent configuration file, you must also make sure the IAM roles of
the sending and target accounts have certain policies. The roles in both the sending and target accounts
should have **CloudWatchAgentServerPolicy**. For more information about assigning this policy to a role,
see Create IAM Roles to Use with CloudWatch Agent on Amazon EC2 Instances (p. 55).

The role in the sending account also must include the following policy. You add this policy in the
Permissions tab in the IAM console when you edit the role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"
            ]
        }
    ]
}
```

The role in the target account must include the following policy, so that it recognizes the IAM role used
by the sending account. You add this policy in the **Trust relationships** tab in the IAM console when
you edit the role. The role in the target account where you add this policy is the role you created in
Create IAM Roles to Use with CloudWatch Agent on Amazon EC2 Instances (p. 55). This role is the role
specified in *agent-role-in-target-account* in the policy used by the sending account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::sending-account-ID:role/role-specified-in-role_arn"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Troubleshooting the CloudWatch Agent

Use the following information to help troubleshoot problems with the CloudWatch agent.

Topics

Installing the CloudWatch Agent Using Run Command Fails

To install the CloudWatch agent using Systems Manager Run Command, the SSM Agent on the target server must be version 2.2.93.0 or later. If your SSM Agent is not the correct version, you may see errors that include the following messages:

```
no latest version found for package AmazonCloudWatchAgent on platform linux
```

```
failed to download installation package reliably
```

For information about updating your SSM Agent version, see Installing and Configuring SSM Agent.

The CloudWatch Agent Won't Start

If the CloudWatch agent fails to start, there might be an issue in your configuration. Configuration information is logged in the **configuration-validation.log** file. This file is located in `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` on Linux servers, and in

```
$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log
```
on servers running Windows Server.

Verify That the CloudWatch Agent is Running

You can query the CloudWatch agent to find whether it is running or stopped.

You can use AWS Systems Manager to do this remotely. You can also use the command line, but only to check the local server.

To query the status of the CloudWatch agent using Run Command

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManageAgent**.
5. In the **Target** area, choose the instance to check.
6. In the **Action** list, choose **status**.
7. Leave **Optional Configuration Source** and **Optional Configuration Location** blank.
8. Choose **Run**.

If the agent is running, the output resembles the following:

```
{
      "status": "running",
      "starttime": "2017-12-12T18:41:18",
      "version": "1.73.4"
}
```

If the agent is stopped the `"status"` field displays `"stopped"`.

To query the status of the CloudWatch agent locally using the command line

- On a Linux server, type the following:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status
  ```

  On a server running Windows Server, type the following in PowerShell as an administrator:

  ```
  & $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2
   -a status
  ```

Where Are the Metrics?

If the CloudWatch agent has been running but you cannot find metrics collected by it in the AWS Management Console or the AWS CLI, confirm that you are using the correct namespace. By default the namespace for metrics collected by the agent is `CWAgent`. You can customize this namespace using the **namespace** field in the **metrics** section of the agent configuration file. If you do not see the metrics that you expect, check the configuration file to confirm the namespace being used.

When you first download the CloudWatch agent package, the agent configuration file is `amazon-cloudwatch-agent.json`. This file is located in the directory where you ran the configuration wizard, or you may have moved it to a different directory. If you use the configuration wizard, the agent configuration file output from the wizard is named `config.json`. For more information about the configuration file, including the **namespace** field, see CloudWatch Agent Configuration File: Metrics Section (p. 87).

Agent Won't Start and the Error Mentions an Amazon EC2 Region

If the agent does not start and the error message mentions an Amazon EC2 region endpoint, you may have configured the agent to need access to the Amazon EC2 endpoint without granting that access.

For example, if you specify a value for the `append_dimensions` parameter in the agent configuration file that depends on Amazon EC2 metadata, and you use proxies, you must make sure that the server can access the endpoint for Amazon EC2. For more information about these endpoints, see Amazon Elastic Compute Cloud (Amazon EC2) in the *Amazon Web Services General Reference*.

CloudWatch Agent Files and Locations

The following table lists the files installed by and used with the CloudWatch agent, along with their locations on servers running Linux or Windows Server.

File	Linux Location	Windows Server Location
The control script that controls starting, stopping, and restarting the agent.	/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl	$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1
The log file the agent writes to. You may need to attach this when contacting customer support.	/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log
Agent configuration validation file.	/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log
The JSON file used to configure the agent, immediately after the wizard creates it. For more information, see Create the CloudWatch Agent Configuration File (p. 81).	/opt/aws/amazon-cloudwatch-agent/bin/config.json	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\config.json
The JSON file used to configure the agent, if this configuration file has been downloaded from Parameter Store.	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json
TOML file used to specify region and credential information to be used by the agent, overriding system defaults.	/opt/aws/amazon-cloudwatch-agent/etc/common-config.toml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.toml

File	Linux Location	Windows Server Location
TOML file generated from the JSON configuration file when the CloudWatch agent is started.	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.toml

Logs Generated by the CloudWatch Agent

The agent generates a log while it runs. This log includes troubleshooting information. This log is the **amazon-cloudwatch-agent.log** file. This file is located in `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` on Linux servers, and in `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log` on servers running Windows Server.

You can configure the agent to log additional details in the **amazon-cloudwatch-agent.log** file. In the agent configuration file, in the **agent** section, set the **debug** field to **true**, then reconfigure and restart the CloudWatch agent. To disable the logging of this extra information, set the **debug** field to **false** reconfigure and restart the agent. For more information, see Manually Create or Edit the CloudWatch Agent Configuration File (p. 86).

Stopping and Restarting the CloudWatch Agent

You can manually stop the CloudWatch agent using either AWS Systems Manager or the command line. When you stop it manually, you also prevent it from automatically starting at the system reboot.

To stop the CloudWatch agent using Run Command

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManageAgent**.
5. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
6. In the **Action** list, choose **stop**.
7. Leave **Optional Configuration Source** and **Optional Configuration Location** blank.
8. Choose **Run**.

To stop the CloudWatch agent locally using the command line

- On a Linux server, type the following:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a stop
```

  On a server running Windows Server, type the following in PowerShell as an administrator:

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2 -a stop
```

To restart the agent, follow the instructions in .

Starting the CloudWatch Agent at System Startup

On both Linux and Windows Server, the CloudWatch agent installation scripts register the agent as a system service. This service is configured to start when your instance or server boots, if the agent was running before the reboot occurred. The CloudWatch agent relies on having been previously configured for it to properly start. If you are preparing an AMI or machine image that starts the CloudWatch agent on newly launched Amazon EC2 instances or on-premises servers, including instances launched by Amazon EC2 Auto Scaling and Spot Fleet, ensure that the `amazon-cloudwatch-agent.toml` and `common-config.toml` files are included in your AMI. These files are located in `/opt/aws/amazon-cloudwatch-agent/etc/` on Linux servers, and in `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\` on servers running Windows Server.

If your instance starts and these files are not present, the CloudWatch agent does not start correctly.

Amazon CloudWatch Metrics and Dimensions Reference

This reference includes all the namespaces, dimensions, and metrics that you can use with CloudWatch. Namespaces are containers for metrics. Metrics, which are time-ordered sets of data points, are isolated from one another in different namespaces so that metrics from different applications are not mistakenly aggregated into the same statistics. In addition, each metric has a dimension, which is a name/value pair that you can use to filter metrics.

Metrics and Dimensions

AWS Namespaces

CloudWatch namespaces are containers for metrics. Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics. All AWS services that provide Amazon CloudWatch data use a namespace string, beginning with "AWS/". When you create custom metrics, you must also specify a namespace as a container for custom metrics. Namespaces for custom metrics cannot start with "AWS/". The following services push metric data points to CloudWatch.

AWS Product	Namespace
Amazon API Gateway	`AWS/ApiGateway`
AppStream 2.0	`AWS/AppStream`
Amazon EC2 Auto Scaling	`AWS/AutoScaling`
AWS Billing	`AWS/Billing`
Amazon CloudFront	`AWS/CloudFront`
Amazon CloudSearch	`AWS/CloudSearch`
Amazon CloudWatch Events	`AWS/Events`
Amazon CloudWatch Logs	`AWS/Logs`
AWS CodeBuild	`AWS/CodeBuild`
Amazon Connect	`AWS/Connect`
AWS Database Migration Service	`AWS/DMS`
AWS Direct Connect	`AWS/DX`
Amazon DynamoDB	`AWS/DynamoDB`
Amazon EC2	`AWS/EC2`
Amazon EC2	`AWS/EC2Spot` (Spot Instances)
Amazon Elastic Container Service	`AWS/ECS`
AWS Elastic Beanstalk	`AWS/ElasticBeanstalk`
Amazon Elastic Block Store	`AWS/EBS`
Amazon Elastic File System	`AWS/EFS`
Elastic Load Balancing	`AWS/ELB` (Classic Load Balancers)
Elastic Load Balancing	`AWS/ApplicationELB` (Application Load Balancers)
Elastic Load Balancing	`AWS/NetworkELB` (Network Load Balancers)
Amazon Elastic Transcoder	`AWS/ElasticTranscoder`
Amazon ElastiCache	`AWS/ElastiCache`
Amazon Elasticsearch Service	`AWS/ES`
Amazon EMR	`AWS/ElasticMapReduce`
Amazon GameLift	`AWS/GameLift`
AWS Glue	`AWS/Glue`
Amazon Inspector	`AWS/Inspector`
AWS IoT	`AWS/IoT`

AWS Product	Namespace
AWS IoT Analytics	`AWS/IoTAnalytics`
AWS Key Management Service	`AWS/KMS`
Amazon Kinesis Data Analytics	`AWS/KinesisAnalytics`
Amazon Kinesis Data Firehose	`AWS/Firehose`
Amazon Kinesis Data Streams	`AWS/Kinesis`
Amazon Kinesis Video Streams	`AWS/KinesisVideo`
AWS Lambda	`AWS/Lambda`
Amazon Lex	`AWS/Lex`
Amazon Machine Learning	`AWS/ML`
Amazon MQ	`AWS/AmazonMQ`
Amazon Neptune	`AWS/Neptune`
AWS OpsWorks	`AWS/OpsWorks`
Amazon Polly	`AWS/Polly`
Amazon Redshift	`AWS/Redshift`
Amazon Relational Database Service	`AWS/RDS`
Amazon Route 53	`AWS/Route53`
Amazon SageMaker	`AWS/SageMaker`
AWS Shield Advanced	`AWS/DDoSProtection`
Amazon Simple Email Service	`AWS/SES`
Amazon Simple Notification Service	`AWS/SNS`
Amazon Simple Queue Service	`AWS/SQS`
Amazon Simple Storage Service	`AWS/S3`
Amazon Simple Workflow Service	`AWS/SWF`
AWS Step Functions	`AWS/States`
AWS Storage Gateway	`AWS/StorageGateway`
Amazon Translate	`AWS/Translate`
AWS Trusted Advisor	`AWS/TrustedAdvisor`
Amazon VPC	`AWS/NATGateway` (NAT gateway)
Amazon VPC	`AWS/VPN` (VPN)
AWS WAF	`WAF`
Amazon WorkSpaces	`AWS/WorkSpaces`

Amazon API Gateway Metrics and Dimensions

The metrics and dimensions that API Gateway sends to Amazon CloudWatch are listed below. For more information, see Monitor API Execution with Amazon CloudWatch in the *Amazon API Gateway Developer Guide*.

API Gateway Metrics

Amazon API Gateway sends metric data to CloudWatch every minute.

The `AWS/ApiGateway` namespace includes the following metrics.

Metric	Description
4XXError	The number of client-side errors captured in a specified period.
	The `Sum` statistic represents this metric, namely, the total count of the 4XXError errors in the given period. The `Average` statistic represents the 4XXError error rate, namely, the total count of the 4XXError errors divided by the total number of requests during the period. The denominator corresponds to the Count metric (below).
	Unit: Count
5XXError	The number of server-side errors captured in a given period.
	The `Sum` statistic represents this metric, namely, the total count of the 5XXError errors in the given period. The `Average` statistic represents the 5XXError error rate, namely, the total count of the 5XXError errors divided by the total number of requests during the period. The denominator corresponds to the Count metric (below).
	Unit: Count
CacheHitCount	The number of requests served from the API cache in a given period.
	The `Sum` statistic represents this metric, namely, the total count of the cache hits in the specified period. The `Average` statistic represents the cache hit rate, namely, the total count of the cache hits divided by the total number of requests during the period. The denominator corresponds to the Count metric (below).
	Unit: Count
CacheMissCount	The number of requests served from the back end in a given period, when API caching is enabled.
	The `Sum` statistic represents this metric, namely, the total count of the cache misses in the specified period.

Metric	Description
	The `Average` statistic represents the cache miss rate, namely, the total count of the cache hits divided by the total number of requests during the period. The denominator corresponds to the Count metric (below). Unit: Count
Count	The total number API requests in a given period. The `SampleCount` statistic represents this metric. Unit: Count
IntegrationLatency	The time between when API Gateway relays a request to the back end and when it receives a response from the back end. Unit: Millisecond
Latency	The time between when API Gateway receives a request from a client and when it returns a response to the client. The latency includes the integration latency and other API Gateway overhead. Unit: Millisecond

Dimensions for Metrics

You can use the dimensions in the following table to filter API Gateway metrics.

Dimension	Description
ApiName	Filters API Gateway metrics for an API of the specified API name.
ApiName, Method, Resource, Stage	Filters API Gateway metrics for an API method of the specified API, stage, resource, and method. API Gateway will not send such metrics unless you have explicitly enabled detailed CloudWatch metrics. You can do this in the console by selecting **Enable CloudWatch Metrics** under a stage **Settings** tab. Alternatively, you can call the stage:update action of the API Gateway REST API to update the `metricsEnabled` property to `true`. Enabling such metrics will incur additional charges to your account. For pricing information, see Amazon CloudWatch Pricing.
ApiName, Stage	Filters API Gateway metrics for an API stage of the specified API and stage.

AppStream 2.0 Metrics and Dimensions

The metrics and dimensions that AppStream 2.0 sends to Amazon CloudWatch are listed below. For more information, see Monitor Amazon AppStream 2.0 With Amazon CloudWatch in the *Amazon AppStream 2.0 Developer Guide*.

Amazon AppStream 2.0 Metrics

AppStream 2.0 sends metrics to CloudWatch one time every minute. The `AWS/AppStream` namespace includes the following metrics.

Metric	Description
ActualCapacity	The total number of instances that are available for streaming or are currently streaming. `ActualCapacity = AvailableCapacity + InUseCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
AvailableCapacity	The number of idle instances currently available for user sessions. `AvailableCapacity = ActualCapacity – InUseCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
CapacityUtilization	The percentage of instances in a fleet that are being used, using the following formula. `CapacityUtilization = (InUseCapacity/ActualCapacity) * 100` Monitoring this metric helps with decisions about increasing or decreasing the value of a fleet's desired capacity. Units: Percent Valid statistics: Average, Minimum, Maximum
DesiredCapacity	The total number of instances that are either running or pending. This represents the total number of concurrent streaming sessions your fleet can support in a steady state. `DesiredCapacity = ActualCapacity + PendingCapacity` Units: Count Valid statistics: Average, Minimum, Maximum
InUseCapacity	The number of instances currently being used for streaming sessions. One `InUseCapacity` count represents one streaming session.

Metric	Description
	Units: Count
	Valid statistics: Average, Minimum, Maximum
`PendingCapacity`	The number of instances being provisioned by AppStream 2.0. Represents the additional number of streaming sessions the fleet can support after provisioning is complete. When provisioning starts, it usually takes 10-20 minutes for an instance to become available for streaming.
	Units: Count
	Valid statistics: Average, Minimum, Maximum
`RunningCapacity`	The total number of instances currently running. Represents the number of concurrent streaming sessions that can be supported by the fleet in its current state.
	This metric is provided for Always-On fleets only, and has the same value as the `ActualCapacity` metric.
	Units: Count
	Valid statistics: Average, Minimum, Maximum
`InsufficientCapacityError`	The number of session requests rejected due to lack of capacity.
	You can set alarms to use this metric to be notified of users waiting for streaming sessions.
	Units: Count
	Valid statistics: Average, Minimum, Maximum, Sum

Dimensions for Amazon AppStream 2.0 Metrics

To filter the metrics provided by Amazon AppStream 2.0, use the following dimension.

Dimension	Description
`Fleet`	The name of the fleet.

Amazon EC2 Auto Scaling Metrics and Dimensions

Amazon EC2 Auto Scaling sends metrics for instances and groups to CloudWatch. For Auto Scaling instances, you can enable detailed (one-minute) monitoring or basic (five-minute) monitoring. For Auto Scaling groups, you can enable group metrics. For more information, see Monitoring Your Auto Scaling Instances and Groups in the *Amazon EC2 Auto Scaling User Guide*.

Auto Scaling Group Metrics

If you enable group metrics, Amazon EC2 Auto Scaling sends aggregated data to CloudWatch every minute.

The `AWS/AutoScaling` namespace includes the following metrics.

Metric	Description
GroupMinSize	The minimum size of the Auto Scaling group.
GroupMaxSize	The maximum size of the Auto Scaling group.
GroupDesiredCapacity	The number of instances that the Auto Scaling group attempts to maintain.
GroupInServiceInstances	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating.
GroupPendingInstances	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating.
GroupStandbyInstances	The number of instances that are in a `Standby` state. Instances in this state are still running but are not actively in service.
GroupTerminatingInstances	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending.
GroupTotalInstances	The total number of instances in the Auto Scaling group. This metric identifies the number of instances that are in service, pending, and terminating.

Dimensions for Auto Scaling Group Metrics

To filter the metrics for your Auto Scaling group by group name, use the `AutoScalingGroupName` dimension.

AWS Billing and Cost Management Dimensions and Metrics

The AWS Billing and Cost Management service sends metrics to CloudWatch. For more information, see Monitoring Charges with Alerts and Notifications in the *AWS Billing and Cost Management User Guide*.

AWS Billing and Cost Management Metrics

The `AWS/Billing` namespace includes the following metrics.

Metric	Description
EstimatedCharges	The estimated charges for your AWS usage. This can either be estimated charges for one service or a roll-up of estimated charges for all services.

Dimensions for AWS Billing and Cost Management Metrics

Billing and Cost Management supports filtering metrics by the following dimensions.

Dimension	Description
`ServiceName`	The name of the AWS service. This dimension is omitted for the total of estimated charges across all services.
`LinkedAccount`	The linked account number. This is used for consolidated billing only. This dimension is included only for accounts that are linked to a separate paying account in a consolidated billing relationship. It is not included for accounts that are not linked to a consolidated billing paying account.
`Currency`	The monetary currency to bill the account. This dimension is required. Unit: USD

Amazon CloudFront Metrics and Dimensions

Amazon CloudFront sends metrics to Amazon CloudWatch for web distributions. Metrics and dimensions are not available for RTMP distributions. For more information, see Monitoring CloudFront Activity Using CloudWatch in the *Amazon CloudFront Developer Guide*.

Amazon CloudFront Metrics

The `AWS/CloudFront` namespace includes the following metrics.

> **Note**
> Only one statistic, Average or Sum, is applicable for each metric. However, all statistics are available through the console, API, and AWS Command Line Interface. In the following table, each metric specifies the statistic that is applicable to that metric.

Metric	Description
`Requests`	The number of requests for all HTTP methods and for both HTTP and HTTPS requests. Valid Statistics: Sum Units: None
`BytesDownloaded`	The number of bytes downloaded by viewers for `GET`, `HEAD`, and `OPTIONS` requests. Valid Statistics: Sum Units: None
`BytesUploaded`	The number of bytes uploaded to your origin with CloudFront using `POST` and `PUT` requests.

Metric	Description
	Valid Statistics: Sum
	Units: None
`TotalErrorRate`	The percentage of all requests for which the HTTP status code is `4xx` or `5xx`.
	Valid Statistics: Average
	Units: Percent
`4xxErrorRate`	The percentage of all requests for which the HTTP status code is `4xx`.
	Valid Statistics: Average
	Units: Percent
`5xxErrorRate`	The percentage of all requests for which the HTTP status code is `5xx`.
	Valid Statistics: Average
	Units: Percent

Dimensions for CloudFront Metrics

CloudFront metrics use the CloudFront namespace and provide metrics for two dimensions:

Dimension	Description
`DistributionId`	The CloudFront ID of the distribution for which you want to display metrics.
`Region`	The region for which you want to display metrics. This value must be `Global`. The `Region` dimension is different from the region in which CloudFront metrics are stored, which is US East (N. Virginia).

Amazon CloudSearch Metrics and Dimensions

Amazon CloudSearch sends metrics to Amazon CloudWatch. For more information, see Monitoring an Amazon CloudSearch Domain with Amazon CloudWatch in the *Amazon CloudSearch Developer Guide*.

Amazon CloudSearch Metrics

The `AWS/CloudSearch` namespace includes the following metrics.

Metric	Description
`SuccessfulRequests`	The number of search requests successfully processed by a search instance.
	Units: Count

Metric	Description
	Valid statistics: Maximum, Sum
SearchableDocuments	The number of searchable documents in the domain's search index. Units: Count Valid statistics: Maximum
IndexUtilization	The percentage of the search instance's index capacity that has been used. The Maximum value indicates the percentage of the domain's index capacity that has been used. Units: Percent Valid statistics: Average, Maximum
Partitions	The number of partitions the index is distributed across. Units: Count Valid statistics: Minimum, Maximum

Dimensions for Amazon CloudSearch Metrics

Amazon CloudSearch sends the ClientId and DomainName dimensions to CloudWatch.

Dimension	Description
ClientId	The AWS account ID.
DomainName	The name of the search domain.

Metrics Collected by the CloudWatch Agent

You can collect metrics from servers by installing the CloudWatch agent on the server. You can install the agent on both Amazon EC2 instances and on-premises servers, and on servers running either Linux or Windows Server. If you install the agent on an Amazon EC2 instance, the metrics it collects are in addition to the metrics enabled by default on Amazon EC2 instances, which are listed in Amazon EC2 Metrics and Dimensions (p. 152).

For information about installing the CloudWatch agent on an instance, see Collect Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent (p. 53).

Metrics Collected by the CloudWatch Agent on Windows Server Instances

On a server running Windows Server, installing the CloudWatch agent enables you to collect the metrics associated with the counters in Windows Performance Monitor. The CloudWatch metric names for these counters are created by putting a space between the object name and the counter name. For example, the `% Interrupt Time` counter of the `Processor` object is given the metric name `Processor % Interrupt Time` in CloudWatch. For more information about Windows Performance Monitor counters, see the Microsoft Windows Server documentation.

The default namespace for metrics collected by the CloudWatch agent is `CWAgent`, although you can specify a different namespace when you configure the agent.

Metrics Collected by the CloudWatch Agent on Linux Instances

The metrics that you can collect with the CloudWatch agent on Linux instances are listed in the following table.

Metric	Description
cpu_time_active	The amount of time that the CPU is active in any capacity. This metric is measured in hundredths of a second. Unit: None
cpu_time_guest	The amount of time that the CPU is running a virtual CPU for a guest operating system. This metric is measured in hundredths of a second. Unit: None
cpu_time_guest_nice	The amount of time that the CPU is running a virtual CPU for a guest operating system which is low-priority and can be interrupted by other processes. This metric is measured in hundredths of a second. Unit: None
cpu_time_idle	The amount of time that the CPU is idle. This metric is measured in hundredths of a second. Unit: None
cpu_time_iowait	The amount of time that the CPU is waiting for I/O operations to complete. This metric is measured in hundredths of a second. Unit: None
cpu_time_irq	The amount of time that the CPU is servicing interrupts. This metric is measured in hundredths of a second. Unit: None
cpu_time_nice	The amount of time that the CPU is in user mode with low-priority processes which can easily be interrupted by higher-priority processes. This metric is measured in hundredths of a second. Unit: None
cpu_time_softirq	The amount of time that the CPU is servicing software interrupts. This metric is measured in hundredths of a second. Unit: None

Metric	Description
cpu_time_steal	The amount of time that the CPU is in *stolen time*, which is time spent in other operating systems in a virtualized environment. This metric is measured in hundredths of a second. Unit: None
cpu_time_system	The amount of time that the CPU is in system mode. This metric is measured in hundredths of a second. Unit: None
cpu_time_user	The amount of time that the CPU is in user mode. This metric is measured in hundredths of a second. Unit: None
cpu_usage_active	The percentage of time that the CPU is active in any capacity. Unit: Percent
cpu_usage_guest	The percentage of time that the CPU is running a virtual CPU for a guest operating system. Unit: Percent
cpu_usage_guest_nice	The percentage of time that the CPU is running a virtual CPU for a guest operating system which is low-priority and can be interrupted by other processes. Unit: Percent
cpu_usage_idle	The percentage of time that the CPU is idle. Unit: Percent
cpu_usage_iowait	The percentage of time that the CPU is waiting for I/O operations to complete. Unit: Percent
cpu_usage_irq	The percentage of time that the CPU is servicing interrupts. Unit: Percent
cpu_usage_nice	The percentage of time that the CPU is in user mode with low-priority processes which can easily be interrupted by higher-priority processes. Unit: Percent
cpu_usage_softirq	The percentage of time that the CPU is servicing software interrupts. Unit: Percent

Metric	Description
cpu_usage_steal	The percentage of time that the CPU is in *stolen time*, which is time spent in other operating systems in a virtualized environment. Unit: Percent
cpu_usage_system	The percentage of time that the CPU is in system mode. Unit: Percent
cpu_usage_user	The percentage of time that the CPU is in user mode. Unit: Percent
disk_free	Free space on the disks. Unit: Bytes
disk_inodes_free	The number of available index nodes on the disk. Unit: Count
disk_inodes_total	The total number of index nodes reserved on the disk. Unit: Count
disk_inodes_used	The number of used index nodes on the disk. Unit: Count
disk_total	Total space on the disks, including used and free. Unit: Bytes
disk_used	Used space on the disks. Unit: Bytes
disk_used_percent	The percentage of total disk space that is used. Unit: Percent
diskio_iops_in_progress	The number of I/O requests that have been issued to the device driver but have not yet completed. Unit: Count
diskio_io_time	The amount of time that the disk has had I/O requests queued. Unit: Milliseconds
diskio_reads	The number of disk read operations. Unit: Count
diskio_read_bytes	The number of bytes read from the disks. Unit: Bytes

Metric	Description
diskio_read_time	The amount of time that read requests have waited on the disks. Multiple read requests waiting at the same time all increase the number. For example, if 5 requests all wait for an average of 100 milliseconds, then 500 is reported. Unit: Milliseconds
diskio_writes	The number disk write operations. Unit: Count
diskio_write_bytes	The number of bytes written to the disks. Unit: Bytes
diskio_write_time	The amount of time that write requests have waited on the disks. Multiple write requests waiting at the same time all increase the number. For example, if 8 requests all wait for an average of 1000 milliseconds, then 8000 is reported. Unit: Milliseconds
mem_active	The amount of memory that has been used in some way during the last sample period. Unit: Bytes
mem_available	The amount of memory that is available and can be given instantly to processes. Unit: Bytes
mem_available_percent	The percentage of memory that is available and can be given instantly to processes. Unit: Percent
mem_buffered	The amount of memory that is being used for buffers. Unit: Bytes
mem_cached	The amount of memory that is being used for file caches. Unit: Bytes
mem_free	The amount of memory that is not being used. Unit: Bytes
mem_inactive	The amount of memory that has not been used in some way during the last sample period Unit: Bytes

Metric	Description
mem_total	The total amount of memory. Unit: Bytes
mem_used	The amount of memory currently in use. Unit: Bytes
mem_used_percent	The percentage of memory currently in use. Unit: Percent
net_bytes_recv	The number of bytes received by the network interface. Unit: Bytes
net_bytes_sent	The number of bytes sent by the network interface. Unit: Bytes
net_drop_in	The number of packets received by this network interface which were dropped. Unit: Count
net_drop_out	The number of packets transmitted by this network interface which were dropped. Unit: Count
net_err_in	The number of receive errors detected by this network interface. Unit: Count
net_err_out	The number of transmit errors detected by this network interface. Unit: Count
net_packets_sent	The number of packets sent by this network interface. Unit: Count
net_packets_recv	The number of packets received by this network interface. Unit: Count
netstat_tcp_close	The number of TCP connections with no state. Unit: Count
netstat_tcp_close_wait	The number of TCP connections waiting for a termination request from the client. Unit: Count

Metric	Description
netstat_tcp_closing	The number of TCP connections that are waiting for a termination request with acknowledgement from the client. Unit: Count
netstat_tcp_established	The number of TCP connections established. Unit: Count
netstat_tcp_fin_wait1	The number of TCP connections in the FIN_WAIT1 state, during the process of closing a connection. Unit: Count
netstat_tcp_fin_wait2	The number of TCP connections in the FIN_WAIT2 state, during the process of closing a connection. Unit: Count
netstat_tcp_last_ack	The number of TCP connections waiting for the client to send acknowledgement of the connection termination message. This is the last state right before the connection is closed down. Unit: Count
netstat_tcp_listen	The number of TCP ports currently listening for a connection request. Unit: Count
netstat_tcp_none	The number of TCP connections with inactive clients. Unit: Count
netstat_tcp_syn_sent	The number of TCP connections waiting for a matching connection request after having sent a connection request. Unit: Count
netstat_tcp_syn_recv	The number of TCP connections waiting for connection request acknowledgement after having sent and received a connection request. Unit: Count
netstat_tcp_time_wait	The number of TCP connections currently waiting to ensure the client received the acknowledgement of its connection termination request. Unit: Count
netstat_udp_socket	The number of current UDP connections. Unit: Count

Metric	Description
processes_blocked	The number of processes that are blocked. Unit: Count
processes_dead	The number of processes that are "dead," which is indicated by the X state code on Linux. Unit: Count
processes_idle	The number of processes that are idle (sleeping for more than 20 seconds). Available only on FreeBSD instances. Unit: Count
processes_paging	The number of processes that are paging, which is indicated by the W state code on Linux. Unit: Count
processes_running	The number of processes that are running, indicated by the R state code. Unit: Count
processes_sleeping	The number of processes that are sleeping, indicated by the S state code. Unit: Count
processes_stopped	The number of processes that are stopped, indicated by the T state code. Unit: Count
processes_total	The total number of processes on the instance. Unit: Count
processes_total_threads	The total number of threads making up the processes. This metric is available only on Linux instances. Unit: Count
processes_wait	The number of processes that are paging, which is indicated by the W state code on FreeBSD instances. This metric is available only on FreeBSD instances. Unit: Count
processes_zombies	The number of zombie processes, indicated by the Z state code. Unit: Count
swap_free	The amount of swap space that is not being used. Unit: Bytes

Metric	Description
swap_used	The amount of swap space currently in use. Unit: Bytes
swap_used_percent	The percentage of swap space currently in use. Unit: Percent

Amazon CloudWatch Events Metrics and Dimensions

CloudWatch Events sends metrics to Amazon CloudWatch every minute.

CloudWatch Events Metrics

The `AWS/Events` namespace includes the following metrics.

All of these metrics use Count as the unit, so Sum and SampleCount are the most useful statistics.

Metric	Description
DeadLetterInvocations	Measures the number of times a rule's target is not invoked in response to an event. This includes invocations that would result in triggering the same rule again, causing an infinite loop. Valid Dimensions: RuleName Units: Count
Invocations	Measures the number of times a target is invoked for a rule in response to an event. This includes successful and failed invocations, but does not include throttled or retried attempts until they fail permanently. It does not include DeadLetterInvocations. **Note** CloudWatch Events only sends this metric to CloudWatch if it has a non-zero value. Valid Dimensions: RuleName Units: Count
FailedInvocations	Measures the number of invocations that failed permanently. This does not include invocations that are retried, or that succeeded after a retry attempt. It also does not count failed invocations that are counted in DeadLetterInvocations. Valid Dimensions: RuleName Units: Count
TriggeredRules	Measures the number of triggered rules that matched with any event. Valid Dimensions: RuleName

Metric	Description
	Units: Count
`MatchedEvents`	Measures the number of events that matched with any rule. Valid Dimensions: None Units: Count
`ThrottledRules`	Measures the number of triggered rules that are being throttled. Valid Dimensions: RuleName Units: Count

Dimensions for CloudWatch Events Metrics

CloudWatch Events metrics have one dimension, which is listed below.

Dimension	Description
`RuleName`	Filters the available metrics by rule name.

Amazon CloudWatch Logs Metrics and Dimensions

CloudWatch Logs sends metrics to CloudWatch every minute.

CloudWatch Logs Metrics

The `AWS/Logs` namespace includes the following metrics.

Metric	Description
`IncomingBytes`	The volume of log events in uncompressed bytes uploaded to CloudWatch Logs. When used with the `LogGroupName` dimension, this is the volume of log events in uncompressed bytes uploaded to the log group. Valid Dimensions: LogGroupName Valid Statistic: Sum Units: Bytes
`IncomingLogEvents`	The number of log events uploaded to CloudWatch Logs. When used with the `LogGroupName` dimension, this is the number of log events uploaded to the log group. Valid Dimensions: LogGroupName Valid Statistic: Sum Units: None

Metric	Description
`ForwardedBytes`	The volume of log events in compressed bytes forwarded to the subscription destination. Valid Dimensions: LogGroupName, DestinationType, FilterName Valid Statistic: Sum Units: Bytes
`ForwardedLogEvents`	The number of log events forwarded to the subscription destination. Valid Dimensions: LogGroupName, DestinationType, FilterName Valid Statistic: Sum Units: None
`DeliveryErrors`	The number of log events for which CloudWatch Logs received an error when forwarding data to the subscription destination. Valid Dimensions: LogGroupName, DestinationType, FilterName Valid Statistic: Sum Units: None
`DeliveryThrottling`	The number of log events for which CloudWatch Logs was throttled when forwarding data to the subscription destination. Valid Dimensions: LogGroupName, DestinationType, FilterName Valid Statistic: Sum Units: None

Dimensions for CloudWatch Logs Metrics

The dimensions that you can use with CloudWatch Logs metrics are listed below.

Dimension	Description
`LogGroupName`	The name of the CloudWatch Logs log group for which to display metrics.
`DestinationType`	The subscription destination for the CloudWatch Logs data, which can be AWS Lambda, Amazon Kinesis Data Streams, or Amazon Kinesis Data Firehose.
`FilterName`	The name of the subscription filter that is forwarding data from the log group to the destination. The subscription filter name is automatically converted by CloudWatch to ASCII and any unsupported characters get replaced with a question mark (?).

AWS CodeBuild Metrics

AWS CodeBuild sends metrics to CloudWatch. For more information, see Monitoring AWS CodeBuild in the *AWS CodeBuild User Guide.*

AWS CodeBuild CloudWatch Metrics

The following metrics can be tracked per AWS account, or per AWS build project.

Metric	Description
Builds	Measures the number of builds triggered. Units: Count Valid CloudWatch statistics: Sum
Duration	Measures the duration of all builds over time. Units: Seconds Valid CloudWatch statistics: Average (recommended), Maximum, Minimum
SucceededBuilds	Measures the number of successful builds. Units: Count Valid CloudWatch statistics: Sum
FailedBuilds	Measures the number of builds that failed because of client error or because of a timeout. Units: Count Valid CloudWatch statistics: Sum

AWS CodeBuild CloudWatch Dimensions

`ProjectName` is the only AWS CodeBuild metrics dimension. If it is specified, then the metrics are for that project. If it is not specified, then the metrics are for the current AWS account.

Dimension	Description
ProjectName	If specified, then the metrics are for that project. If not specified, then the metrics are for the AWS account.

Amazon Connect Metrics

For information about the Amazon Connect metrics that you can use with CloudWatch, see Monitoring Amazon Connect Using Amazon CloudWatch Metrics in the Amazon Connect Administrator Guide.

AWS DMS Metrics

For information about the AWS DMS metrics that you can use with CloudWatch, see Monitoring AWS Database Migration Service Tasks in the AWS Database Migration Service User Guide.

AWS Direct Connect Metrics and Dimensions

By default, CloudWatch provides AWS Direct Connect metric data in 5-minute intervals. You can optionally view data in 1-minute intervals. For more information, see Monitoring with CloudWatch in the *AWS Direct Connect User Guide*.

AWS Direct Connect Metrics

The following metrics are available from AWS Direct Connect. Metrics are currently available for AWS Direct Connect physical connections only.

Metric	Description
ConnectionState	The state of the connection. 0 indicates DOWN and 1 indicates UP. Units: Boolean
ConnectionBpsEgress	The bit rate for outbound data from the AWS side of the connection. The number reported is the aggregate over the specified time period (5 minutes by default, 1 minute minimum). Units: Bits per second
ConnectionBpsIngress	The bit rate for inbound data to the AWS side of the connection. The number reported is the aggregate over the specified time period (5 minutes by default, 1 minute minimum). Units: Bits per second
ConnectionPpsEgress	The packet rate for outbound data from the AWS side of the connection. The number reported is the aggregate over the specified time period (5 minutes by default, 1 minute minimum). Units: Packets per second
ConnectionPpsIngress	The packet rate for inbound data to the AWS side of the connection. The number reported is the aggregate over the specified time period (5 minutes by default, 1 minute minimum).

Metric	Description
	Units: Packets per second
ConnectionCRCErrorCount	The number of times cyclic redundancy check (CRC) errors are observed for the data received at the connection. Units: Integer
ConnectionLightLevelTx	Indicates the health of the fiber connection for egress (outbound) traffic from the AWS side of the connection. This metric is available for connections with 10 Gbps port speeds only. Units: dBm
ConnectionLightLevelRx	Indicates the health of the fiber connection for ingress (inbound) traffic to the AWS side of the connection. This metric is available for connections with 10 Gbps port speeds only. Units: dBm

Dimensions for AWS Direct Connect Metrics

You can filter the AWS Direct Connect data using the following dimensions.

Dimension	Description
ConnectionId	This dimension filters the data by the AWS Direct Connect connection.

Amazon DynamoDB Metrics and Dimensions

Amazon DynamoDB sends metrics to CloudWatch. For more information, see Monitoring DynamoDB Tables with Amazon CloudWatch in the *Amazon DynamoDB Developer Guide*.

DynamoDB Metrics

The following metrics are available from Amazon DynamoDB. Note that DynamoDB only sends metrics to CloudWatch when they have a non-zero value. For example, the `UserErrors` metric is incremented whenever a request generates an HTTP 400 status code. If no HTTP 400 errors were encountered during a time period, CloudWatch will not provide metrics for `UserErrors` during that period.

Note
Amazon CloudWatch aggregates the following DynamoDB metrics at one-minute intervals:

- `ConditionalCheckFailedRequests`
- `ConsumedReadCapacityUnits`
- `ConsumedWriteCapacityUnits`

- `ReadThrottleEvents`
- `ReturnedBytes`
- `ReturnedItemCount`
- `ReturnedRecordsCount`
- `SuccessfulRequestLatency`
- `SystemErrors`
- `TimeToLiveDeletedItemCount`
- `ThrottledRequests`
- `UserErrors`
- `WriteThrottleEvents`

For all other DynamoDB metrics, the aggregation granularity is five minutes.

Not all statistics, such as *Average* or *Sum*, are applicable for every metric. However, all of these values are available through the Amazon DynamoDB console, or by using the CloudWatch console, AWS CLI, or AWS SDKs for all metrics. In the following table, each metric has a list of Valid Statistics that is applicable to that metric.

Metric	Description
`ConditionalCheckFailedRequests`	The number of failed attempts to perform conditional writes. The `PutItem`, `UpdateItem`, and `DeleteItem` operations let you provide a logical condition that must evaluate to true before the operation can proceed. If this condition evaluates to false, `ConditionalCheckFailedRequests` is incremented by one. **Note** A failed conditional write will result in an HTTP 400 error (Bad Request). These events are reflected in the `ConditionalCheckFailedRequests` metric, but not in the `UserErrors` metric. Units: `Count` Dimensions: `TableName` Valid Statistics: - `Minimum` - `Maximum` - `Average` - `SampleCount` - `Sum`
`ConsumedReadCapacityUnits`	The number of read capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. You can retrieve the total consumed read capacity for a table and all of its global secondary indexes, or for a particular global secondary index. For more information, see Provisioned Throughput in Amazon DynamoDB.

Metric	Description
	Note Use the `Sum` statistic to calculate the consumed throughput. For example, get the `Sum` value over a span of one minute, and divide it by the number of seconds in a minute (60) to calculate the average `ConsumedReadCapacityUnits` per second (recognizing that this average will not highlight any large but brief spikes in read activity that occurred during that minute). You can compare the calculated value to the provisioned throughput value you provide DynamoDB. Units: `Count` Dimensions: `TableName, GlobalSecondaryIndexName` Valid Statistics: • `Minimum` – Minimum number of read capacity units consumed by any individual request to the table or index. • `Maximum` – Maximum number of read capacity units consumed by any individual request to the table or index. • `Average` – Average per-request read capacity consumed. • `Sum` – Total read capacity units consumed. This is the most useful statistic for the `ConsumedReadCapacityUnits` metric. • `SampleCount` – Number of requests to DynamoDB that consumed read capacity.

Metric	Description
ConsumedWriteCapacityUnits	The number of write capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. You can retrieve the total consumed write capacity for a table and all of its global secondary indexes, or for a particular global secondary index. For more information, see Provisioned Throughput in Amazon DynamoDB.

The number of write capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. You can retrieve the total consumed write capacity for a table and all of its global secondary indexes, or for a particular global secondary index. For more information, see Provisioned Throughput in Amazon DynamoDB.

> **Note**
> Use the Sum statistic to calculate the consumed throughput. For example, get the Sum value over a span of one minute, and divide it by the number of seconds in a minute (60) to calculate the average ConsumedWriteCapacityUnits per second (recognizing that this average will not highlight any large but brief spikes in write activity that occurred during that minute). You can compare the calculated value to the provisioned throughput value you provide DynamoDB.

Units: Count

Dimensions: TableName, GlobalSecondaryIndexName

Valid Statistics:

- Minimum – Minimum number of write capacity units consumed by any individual request to the table or index.
- Maximum – Maximum number of write capacity units consumed by any individual request to the table or index.
- Average – Average per-request write capacity consumed.
- Sum – Total write capacity units consumed. This is the most useful statistic for the ConsumedWriteCapacityUnits metric.
- SampleCount – Number of requests to DynamoDB that consumed write capacity.

Metric	Description
OnlineIndexConsumedWriteCapacity	The number of write capacity units consumed when adding a new global secondary index to a table. If the write capacity of the index is too low, incoming write activity during the backfill phase might be throttled; this can increase the time it takes to create the index. You should monitor this statistic while the index is being built to determine whether the write capacity of the index is underprovisioned.
	You can adjust the write capacity of the index using the `UpdateTable` operation, even while the index is still being built.
	Note that the `ConsumedWriteCapacityUnits` metric for the index does not include the write throughput consumed during index creation.
	Units: `Count`
	Dimensions: `TableName, GlobalSecondaryIndexName`
	Valid Statistics:
	`Minimum` `Maximum` `Average` `SampleCount` `Sum`
OnlineIndexPercentageProgress	The percentage of completion when a new global secondary index is being added to a table. DynamoDB must first allocate resources for the new index, and then backfill attributes from the table into the index. For large tables, this process might take a long time. You should monitor this statistic to view the relative progress as DynamoDB builds the index.
	Units: `Count`
	Dimensions: `TableName, GlobalSecondaryIndexName`
	Valid Statistics:
	`Minimum` `Maximum` `Average` `SampleCount` `Sum`

Metric	Description
OnlineIndexThrottleEvents	The number of write throttle events that occur when adding a new global secondary index to a table. These events indicate that the index creation will take longer to complete, because incoming write activity is exceeding the provisioned write throughput of the index.
	You can adjust the write capacity of the index using the `UpdateTable` operation, even while the index is still being built.
	Note that the `WriteThrotttleEvents` metric for the index does not include any throttle events that occur during index creation.
	Units: `Count`
	Dimensions: `TableName, GlobalSecondaryIndexName`
	Valid Statistics: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `Sum`
PendingReplicationCount	(This metric is for DynamoDB global tables.) The number of item updates that are written to one replica table, but that have not yet been written to another replica in the global table.
	Units: `Count`
	Dimensions: `TableName, ReceivingRegion`
	Valid Statistics: • `Average` • `Sample Count` • `Sum`

Metric	Description
`ProvisionedReadCapacityUnits`	The number of provisioned read capacity units for a table or a global secondary index. The `TableName` dimension returns the `ProvisionedReadCapacityUnits` for the table, but not for any global secondary indexes. To view `ProvisionedReadCapacityUnits` for a global secondary index, you must specify both `TableName` and `GlobalSecondaryIndex`. Units: `Count` Dimensions: `TableName, GlobalSecondaryIndexName` Valid Statistics: • `Minimum` – Lowest setting for provisioned read capacity. If you use `UpdateTable` to increase read capacity, this metric shows the lowest value of provisioned `ReadCapacityUnits` during this time period. • `Maximum` – Highest setting for provisioned read capacity. If you use `UpdateTable` to decrease read capacity, this metric shows the highest value of provisioned `ReadCapacityUnits` during this time period. • `Average` – Average provisioned read capacity. The `ProvisionedReadCapacityUnits` metric is published at five-minute intervals. Therefore, if you rapidly adjust the provisioned read capacity units, this statistic might not reflect the true average.

Metric	Description
ProvisionedWriteCapacityUnits	The number of provisioned write capacity units for a table or a global secondary index The `TableName` dimension returns the `ProvisionedWriteCapacityUnits` for the table, but not for any global secondary indexes. To view `ProvisionedWriteCapacityUnits` for a global secondary index, you must specify both `TableName` and `GlobalSecondaryIndex`. Units: `Count` Dimensions: `TableName, GlobalSecondaryIndexName` Valid Statistics: • `Minimum` – Lowest setting for provisioned write capacity. If you use `UpdateTable` to increase write capacity, this metric shows the lowest value of provisioned `WriteCapacityUnits` during this time period. • `Maximum` – Highest setting for provisioned write capacity. If you use `UpdateTable` to decrease write capacity, this metric shows the highest value of provisioned `WriteCapacityUnits` during this time period. • `Average` – Average provisioned write capacity. The `ProvisionedWriteCapacityUnits` metric is published at five-minute intervals. Therefore, if you rapidly adjust the provisioned write capacity units, this statistic might not reflect the true average.
ReadThrottleEvents	Requests to DynamoDB that exceed the provisioned read capacity units for a table or a global secondary index. A single request can result in multiple events. For example, a `BatchGetItem` that reads 10 items is processed as ten `GetItem` events. For each event, `ReadThrottleEvents` is incremented by one if that event is throttled. The `ThrottledRequests` metric for the entire `BatchGetItem` is not incremented unless *all ten* of the `GetItem` events are throttled. The `TableName` dimension returns the `ReadThrottleEvents` for the table, but not for any global secondary indexes. To view `ReadThrottleEvents` for a global secondary index, you must specify both `TableName` and `GlobalSecondaryIndex`. Units: `Count` Dimensions: `TableName, GlobalSecondaryIndexName` Valid Statistics: • `SampleCount` • `Sum`

Metric	Description
ReplicationLatency	(This metric is for DynamoDB global tables.) The elapsed time between an updated item appearing in the DynamoDB stream for one replica table, and that item appearing in another replica in the global table. Units: `Milliseconds` Dimensions: `TableName, ReceivingRegion` Valid Statistics: • `Average` • `Minimum` • `Maximum`
ReturnedBytes	The number of bytes returned by `GetRecords` operations (Amazon DynamoDB Streams) during the specified time period. Units: `Bytes` Dimensions: `Operation, StreamLabel, TableName` Valid Statistics: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `Sum`
ReturnedItemCount	The number of items returned by `Query` or `Scan` operations during the specified time period. Note that the number of items *returned* is not necessarily the same as the number of items that were evaluated. For example, suppose you requested a `Scan` on a table that had 100 items, but specified a `FilterExpression` that narrowed the results so that only 15 items were returned. In this case, the response from `Scan` would contain a `ScanCount` of 100 and a `Count` of 15 returned items. Units: `Count` Dimensions: `TableName, Operation` Valid Statistics: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `Sum`

Metric	Description
ReturnedRecordsCount	The number of stream records returned by `GetRecords` operations (Amazon DynamoDB Streams) during the specified time period. Units: `Count` Dimensions: `Operation, StreamLabel, TableName` Valid Statistics: • `Minimum` • `Maximum` • `Average` • `SampleCount` • `Sum`
SuccessfulRequestLatency	Successful requests to DynamoDB or Amazon DynamoDB Streams during the specified time period. `SuccessfulRequestLatency` can provide two different kinds of information: • The elapsed time for successful requests (`Minimum`, `Maximum`, `Sum`, or `Average`). • The number of successful requests (`SampleCount`). `SuccessfulRequestLatency` reflects activity only within DynamoDB or Amazon DynamoDB Streams, and does not take into account network latency or client-side activity. Units: `Milliseconds` Dimensions: `TableName, Operation` Valid Statistics: • `Minimum` • `Maximum` • `Average` • `SampleCount`
SystemErrors	Requests to DynamoDB or Amazon DynamoDB Streams that generate an HTTP 500 status code during the specified time period. An HTTP 500 usually indicates an internal service error. Units: `Count` Dimensions: All dimensions Valid Statistics: • `Sum` • `SampleCount`

Metric	Description
`TimeToLiveDeletedItemCount`	The number of items deleted by Time To Live (TTL) during the specified time period. This metric helps you monitor the rate of TTL deletions on your table. Units: `Count` Dimensions: TableName Valid Statistics: • `Sum`

Metric	Description
`ThrottledRequests`	Requests to DynamoDB that exceed the provisioned throughput limits on a resource (such as a table or an index).
	`ThrottledRequests` is incremented by one if any event within a request exceeds a provisioned throughput limit. For example, if you update an item in a table with global secondary indexes, there are multiple events—a write to the table, and a write to each index. If one or more of these events are throttled, then `ThrottledRequests` is incremented by one.
	Note In a batch request (`BatchGetItem` or `BatchWriteItem`), `ThrottledRequests` is only incremented if *every* request in the batch is throttled. If any individual request within the batch is throttled, one of the following metrics is incremented:
	• `ReadThrottleEvents` – For a throttled `GetItem` event within `BatchGetItem`. • `WriteThrottleEvents` – For a throttled `PutItem` or `DeleteItem` event within `BatchWriteItem`.
	To gain insight into which event is throttling a request, compare `ThrottledRequests` with the `ReadThrottleEvents` and `WriteThrottleEvents` for the table and its indexes.
	Note A throttled request will result in an HTTP 400 status code. All such events are reflected in the `ThrottledRequests` metric, but not in the `UserErrors` metric.
	Units: `Count`
	Dimensions: `TableName, Operation`
	Valid Statistics:
	• `Sum` • `SampleCount`

Metric	Description
UserErrors	Requests to DynamoDB or Amazon DynamoDB Streams that generate an HTTP 400 status code during the specified time period. An HTTP 400 usually indicates a client-side error such as an invalid combination of parameters, attempting to update a nonexistent table, or an incorrect request signature. All such events are reflected in the `UserErrors` metric, except for the following: • *ProvisionedThroughputExceededException* – See the `ThrottledRequests` metric in this section. • *ConditionalCheckFailedException* – See the `ConditionalCheckFailedRequests` metric in this section. `UserErrors` represents the aggregate of HTTP 400 errors for DynamoDB or Amazon DynamoDB Streams requests for the current region and the current AWS account. Units: `Count` Valid Statistics: • `Sum` • `SampleCount`

Metric	Description
WriteThrottleEvents	Requests to DynamoDB that exceed the provisioned write capacity units for a table or a global secondary index.
	A single request can result in multiple events. For example, a `PutItem` request on a table with three global secondary indexes would result in four events—the table write, and each of the three index writes. For each event, the `WriteThrottleEvents` metric is incremented by one if that event is throttled. For single `PutItem` requests, if any of the events are throttled, `ThrottledRequests` is also incremented by one. For `BatchWriteItem`, the `ThrottledRequests` metric for the entire `BatchWriteItem` is not incremented unless all of the individual `PutItem` or `DeleteItem` events are throttled.
	The `TableName` dimension returns the `WriteThrottleEvents` for the table, but not for any global secondary indexes. To view `WriteThrottleEvents` for a global secondary index, you must specify both `TableName` and `GlobalSecondaryIndex`.
	Units: `Count`
	Dimensions: `TableName, GlobalSecondaryIndexName`
	Valid Statistics:
	• `Sum`
	• `SampleCount`

Dimensions for DynamoDB Metrics

The metrics for DynamoDB are qualified by the values for the account, table name, global secondary index name, or operation. You can use the CloudWatch console to retrieve DynamoDB data along any of the dimensions in the table below.

Dimension	Description
GlobalSecondaryIndexName	This dimension limits the data to a global secondary index on a table. If you specify `GlobalSecondaryIndexName`, you must also specify `TableName`.
Operation	This dimension limits the data to one of the following DynamoDB operations:
	• `PutItem`
	• `DeleteItem`
	• `UpdateItem`
	• `GetItem`
	• `BatchGetItem`
	• `Scan`
	• `Query`

Dimension	Description
	• `BatchWriteItem` In addition, you can limit the data to the following Amazon DynamoDB Streams operation: • `GetRecords`
`ReceivingRegion`	This dimension limits the data to a particular AWS region. It is used with metrics originating from replica tables within a DynamoDB global table.
`StreamLabel`	This dimension limits the data to a specific stream label. It is used with metrics originating from Amazon DynamoDB Streams `GetRecords` operations.
`TableName`	This dimension limits the data to a specific table. This value can be any table name in the current region and the current AWS account.

Amazon EC2 Metrics and Dimensions

Amazon Elastic Compute Cloud (Amazon EC2) sends metrics to CloudWatch for your EC2 instances. Basic (five-minute) monitoring is enabled by default. You can enable detailed (one-minute) monitoring. For information about additional metrics for Amazon EC2 instances that are in an Auto Scaling group, see Amazon EC2 Auto Scaling Metrics and Dimensions (p. 121).

For more information about how to monitor Amazon EC2, see Monitoring Your Instances with CloudWatch in the *Amazon EC2 User Guide for Linux Instances*.

Metric Collection and Calculation on C5 and M5 Instances

Because C5 and M5 instances use the Nitro hypervisor, they publish CloudWatch metrics differently than other instances, which use a Xen-based hypervisor. For more information, see Nitro Hypervisor.

In basic monitoring for EC2 instances, seven pre-selected metrics are available at the five-minute frequency. When you use basic monitoring on any type of EC2 instance (using either hypervisor), the hypervisor measures five separate samples per metric during each five-minute interval. The way that these data points are published to CloudWatch depends on the type of hypervisor used by the instance.

- On Xen instances, these five samples are aggregated and reported to CloudWatch as a single data point at the end of the five-minute interval, with a time stamp corresponding to the beginning of the five-minute interval. The statistic set for this data point includes a `SampleCount` of 5.
- On Nitro instances, the data point is published to CloudWatch immediately after the first sample is taken during the interval. When each subsequent sample within the interval occurs, the statistic set for that data point is updated to reflect all the samples taken so far, but the time stamp remains the same. If you are graphing or monitoring the data as it is reported, the statistics for the current data point can appear to change as each sample is taken during the five-minute interval. Each of the five samples taken during the interval are included in the `SampleCount` statistic.

Time	1:01PM	1:02PM	1:03PM	1:04PM	1:05PM

Sample Value	10	15	10	5	10
Published on Xen instances (with basic monitoring enabled)	Nothing yet	Nothing yet	Nothing yet	Nothing yet	Average=10 Sum=50 Minimum=5 Maximum=15 SampleCount=5 Timestamp=1:00PM
Published on Nitro instances (with basic monitoring enabled)	Average=10 Sum=10 Minimum=10 Maximum=10 SampleCount=1 Timestamp=1:00PM	Average=12.5 Sum=25 Minimum=10 Maximum=15 SampleCount=2 Timestamp=1:00PM	Average=11.666 Sum=35 Minimum=10 Maximum=15 SampleCount=3 Timestamp=1:00PM	Average=10 Sum=40 Minimum=5 Maximum=15 SampleCount=4 Timestamp=1:00PM	Average=10 Sum=50 Minimum=5 Maximum=15 SampleCount=5 Timestamp=1:00PM

In the example shown in the preceding table, basic monitoring is being used. A sample is taken each minute over a five-minute interval. On a Xen-based instance, nothing is published until the end of the five-minute interval. At this time (1:05PM), the five samples are aggregated and a single data point is written with a 1:00PM time stamp. This data point has a `SampleCount` of 5. This one data point contributes 50 to the `Sum` statistic for the metric.

On a Nitro instance, the data point is first written to CloudWatch at 1:01PM with a time stamp of 1:00PM and a value of 10. At 1:02PM, the new sample of 15 causes the Average of the 1:00PM data point to change to 12.5, the average of the two samples. This also changes the Maximum to 15. The time stamp of this data point remains 1:00PM. At 1:03PM, the data point Average changes to 11.6. Finally, after the last sample at 1:05PM, the Average returns to 10. The final Average statistic of the data point is 10, but it has contributed 50 to the `Sum` as this five-minute period includes a `SampleCount` of 5 and each of the five samples is included in the `Sum`. The `Minimum` and `Maximum` values reflect the highest and lowest values of the samples taken during the interval.

Because of the way that Nitro instances collect and calculate the data, we recommend that you only set alarms with at least two evaluation periods or require "M out of N" breaching periods, where M is at least 2. This is because on Nitro instances, a single sample during one five-minute period can cause the data point to temporarily breach the threshold, even if subsequent samples in that period bring the final statistic set of this data point to within the threshold. This "mutable" data point in the current sample can cause the alarm to activate during the middle of the interval. For more information, see Evaluating an Alarm (p. 327).

Amazon EC2 Metrics

The following metrics are available by default from each EC2 instance. You can enable additional metrics by installing the CloudWatch agent on the instance. For more information about installing the CloudWatch agent on an instance, see Collect Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent (p. 53). For more information about the additional metrics that can be collected, see Metrics Collected by the CloudWatch Agent (p. 125).

The `AWS/EC2` namespace includes the following CPU credit metrics for your T2 instances.

Metric	Description
CPUCreditUsage	[T3/T2 instances] The number of CPU credits spent by the instance for CPU utilization. One CPU credit equals one vCPU running at 100% utilization for one minute or an equivalent combination of vCPUs, utilization, and time (for example, one vCPU running at 50% utilization for two minutes or two vCPUs running at 25% utilization for two minutes). CPU credit metrics are available at a five-minute frequency only. If you specify a period greater than five minutes, use the Sum statistic instead of the Average statistic. Units: Credits (vCPU-minutes)
CPUCreditBalance	[T3/T2 instances] The number of earned CPU credits that an instance has accrued since it was launched or started. For T2 Standard, the CPUCreditBalance also includes the number of launch credits that have been accrued. Credits are accrued in the credit balance after they are earned, and removed from the credit balance when they are spent. The credit balance has a maximum limit, determined by the instance size. After the limit is reached, any new credits that are earned are discarded. For T2 Standard, launch credits do not count towards the limit. The credits in the CPUCreditBalance are available for the instance to spend to burst beyond its baseline CPU utilization. When an instance is running, credits in the CPUCreditBalance do not expire. When a T3 instance stops, the CPUCreditBalance value persists for seven days. Thereafter, all accrued credits are lost. When a T2 instance stops, the CPUCreditBalance value does not persist, and all accrued credits are lost. CPU credit metrics are available at a five-minute frequency only. Units: Credits (vCPU-minutes)
CPUSurplusCreditBalance	[T3/T2 Unlimited instances] The number of surplus credits that have been spent by a T3 Unlimited or T2 Unlimited instance when its CPUCreditBalance value is zero. The CPUSurplusCreditBalance value is paid down by earned CPU credits. If the number of surplus credits exceeds the maximum number of credits that the instance can earn in a 24-hour period, the spent surplus credits above the maximum incur an additional charge. Units: Credits (vCPU-minutes)
CPUSurplusCreditsCharged	[T3/T2 Unlimited instances] The number of spent surplus credits that are not paid down by earned CPU credits, and which thus incur an additional charge. Spent surplus credits are charged when any of the following occurs:

Metric	Description
	• The spent surplus credits exceed the maximum number of credits that the instance can earn in a 24-hour period. Spent surplus credits above the maximum are charged at the end of the hour. • The instance is stopped or terminated. • The instance is switched from `unlimited` to `standard`. Units: Credits (vCPU-minutes)

The `AWS/EC2` namespace includes the following instance metrics.

Metric	Description
CPUUtilization	The percentage of allocated EC2 compute units that are currently in use on the instance. This metric identifies the processing power required to run an application upon a selected instance. To use the percentiles statistic, you must enable detailed monitoring. Depending on the instance type, tools in your operating system can show a lower percentage than CloudWatch when the instance is not allocated a full processor core. Units: Percent
DiskReadOps	Completed read operations from all instance store volumes available to the instance in a specified period of time. To calculate the average I/O operations per second (IOPS) for the period, divide the total operations in the period by the number of seconds in that period. If there are no instance store volumes, either the value is 0 or the metric is not reported. Units: Count
DiskWriteOps	Completed write operations to all instance store volumes available to the instance in a specified period of time. To calculate the average I/O operations per second (IOPS) for the period, divide the total operations in the period by the number of seconds in that period. If there are no instance store volumes, either the value is 0 or the metric is not reported. Units: Count
DiskReadBytes	Bytes read from all instance store volumes available to the instance. This metric is used to determine the volume of the data the application reads from the hard disk of the instance. This can be used to determine the speed of the application.

Metric	Description
	The number reported is the number of bytes received during the period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to find Bytes/second. If you have detailed (one-minute) monitoring, divide it by 60.
	If there are no instance store volumes, either the value is 0 or the metric is not reported.
	Units: Bytes
DiskWriteBytes	Bytes written to all instance store volumes available to the instance.
	This metric is used to determine the volume of the data the application writes onto the hard disk of the instance. This can be used to determine the speed of the application.
	The number reported is the number of bytes received during the period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to find Bytes/second. If you have detailed (one-minute) monitoring, divide it by 60.
	If there are no instance store volumes, either the value is 0 or the metric is not reported.
	Units: Bytes
NetworkIn	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to a single instance.
	The number reported is the number of bytes received during the period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to find Bytes/second. If you have detailed (one-minute) monitoring, divide it by 60.
	Units: Bytes
NetworkOut	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic from a single instance.
	The number reported is the number of bytes sent during the period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to find Bytes/second. If you have detailed (one-minute) monitoring, divide it by 60.
	Units: Bytes
NetworkPacketsIn	The number of packets received on all network interfaces by the instance. This metric identifies the volume of incoming traffic in terms of the number of packets on a single instance. This metric is available for basic monitoring only.
	Units: Count
	Statistics: Minimum, Maximum, Average

Metric	Description
NetworkPacketsOut	The number of packets sent out on all network interfaces by the instance. This metric identifies the volume of outgoing traffic in terms of the number of packets on a single instance. This metric is available for basic monitoring only.
	Units: Count
	Statistics: Minimum, Maximum, Average

The `AWS/EC2` namespace includes the following status checks metrics. By default, status check metrics are available at a 1-minute frequency at no charge. For a newly-launched instance, status check metric data is only available after the instance has completed the initialization state (within a few minutes of the instance entering the running state). For more information about EC2 status checks, see Status Checks For Your Instances.

Metric	Description
StatusCheckFailed	Reports whether the instance has passed both the instance status check and the system status check in the last minute.
	This metric can be either 0 (passed) or 1 (failed).
	By default, this metric is available at a 1-minute frequency at no charge.
	Units: Count
StatusCheckFailed_Instance	Reports whether the instance has passed the instance status check in the last minute.
	This metric can be either 0 (passed) or 1 (failed).
	By default, this metric is available at a 1-minute frequency at no charge.
	Units: Count
StatusCheckFailed_System	Reports whether the instance has passed the system status check in the last minute.
	This metric can be either 0 (passed) or 1 (failed).
	By default, this metric is available at a 1-minute frequency at no charge.
	Units: Count

Amazon CloudWatch data for a new EC2 instance typically becomes available within one minute of the end of the first period of time requested (the *aggregation period*) in the query. You can set the period—the length of time over which statistics are aggregated—with the Period parameter. For more information on periods, see Periods (p. 6).

You can use the currently available dimensions for EC2 instances (for example, `ImageId` or `InstanceType`) to refine the metrics returned. For information about the dimensions you can use with EC2, see Dimensions for Amazon EC2 Metrics (p. 159).

The `AWS/EC2` namespace includes the following Amazon EBS metrics for your C5 and M5 instances.

Metric	Description
EBSReadOps	Completed read operations from all Amazon EBS volumes attached to the instance in a specified period of time.
	To calculate the average read I/O operations per second (Read IOPS) for the period, divide the total operations in the period by the number of seconds in that period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to calculate the Read IOPS. If you have detailed (one-minute) monitoring, divide it by 60.
	Unit: Count
EBSWriteOps	Completed write operations to all EBS volumes attached to the instance in a specified period of time.
	To calculate the average write I/O operations per second (Write IOPS) for the period, divide the total operations in the period by the number of seconds in that period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to calculate the Write IOPS. If you have detailed (one-minute) monitoring, divide it by 60.
	Unit: Count
EBSReadBytes	Bytes read from all EBS volumes attached to the instance in a specified period of time.
	The number reported is the number of bytes read during the period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to find Read Bytes/second. If you have detailed (one-minute) monitoring, divide it by 60.
	Unit: Bytes
EBSWriteBytes	Bytes written to all EBS volumes attached to the instance in a specified period of time.
	The number reported is the number of bytes written during the period. If you are using basic (five-minute) monitoring, you can divide this number by 300 to find Write Bytes/second. If you have detailed (one-minute) monitoring, divide it by 60.
	Unit: Bytes
EBSIOBalance%	Available only for the smaller C5 and M5 instance sizes. Provides information about the percentage of I/O credits remaining in the burst bucket. This metric is available for basic monitoring only.
	The `Sum` statistic is not applicable to this metric.

Metric	Description
	Unit: Percent
`EBSByteBalance%`	Available only for the smaller C5 and M5 instance sizes. Provides information about the percentage of throughput credits remaining in the burst bucket. This metric is available for basic monitoring only. The `Sum` statistic is not applicable to this metric. Unit: Percent

Dimensions for Amazon EC2 Metrics

If you're using Detailed Monitoring, you can filter the EC2 instance data using any of the dimensions in the following table.

Dimension	Description
`AutoScalingGroupName`	This dimension filters the data you request for all instances in a specified capacity group. An *Auto Scaling group* is a collection of instances you define if you're using Auto Scaling. This dimension is available only for Amazon EC2 metrics when the instances are in such an Auto Scaling group. Available for instances with Detailed or Basic Monitoring enabled.
`ImageId`	This dimension filters the data you request for all instances running this Amazon EC2 Amazon Machine Image (AMI). Available for instances with Detailed Monitoring enabled.
`InstanceId`	This dimension filters the data you request for the identified instance only. This helps you pinpoint an exact instance from which to monitor data.
`InstanceType`	This dimension filters the data you request for all instances running with this specified instance type. This helps you categorize your data by the type of instance running. For example, you might compare data from an m1.small instance and an m1.large instance to determine which has the better business value for your application. Available for instances with Detailed Monitoring enabled.

Amazon EC2 Spot Fleet Metrics and Dimensions

Amazon Elastic Compute Cloud (Amazon EC2) sends information about your Spot fleet to CloudWatch. For more information, see CloudWatch Metrics for Spot Fleet in the *Amazon EC2 User Guide for Linux Instances*.

Amazon EC2 Spot Fleet Metrics

The `AWS/EC2Spot` namespace includes the following metrics, plus the CloudWatch metrics for the Spot instances in your fleet.

The `AWS/EC2Spot` namespace includes the following metrics.

Metric	Description
`AvailableInstancePoolsCount`	The Spot Instance pools specified in the Spot Fleet request. Units: Count
`BidsSubmittedForCapacity`	The capacity for which Amazon EC2 has submitted bids. Units: Count
`EligibleInstancePoolCount`	The Spot Instance pools specified in the Spot Fleet request where Amazon EC2 can fulfill bids. Amazon EC2 will not fulfill bids in pools where your bid price is less than the Spot price or the Spot price is greater than the price for On-Demand instances. Units: Count
`FulfilledCapacity`	The capacity that Amazon EC2 has fulfilled. Units: Count
`MaxPercentCapacityAllocation`	The maximum value of `PercentCapacityAllocation` across all Spot Instance pools specified in the Spot Fleet request. Units: Percent
`PendingCapacity`	The difference between `TargetCapacity` and `FulfilledCapacity`. Units: Count
`PercentCapacityAllocation`	The capacity allocated for the Spot Instance pool for the specified dimensions. To get the maximum value recorded across all Spot Instance pools, use `MaxPercentCapacityAllocation`. Units: Percent
`TargetCapacity`	The target capacity of the Spot Fleet request. Units: Count
`TerminatingCapacity`	The capacity that is being terminated due to Spot Instance interruptions. Units: Count

If the unit of measure for a metric is `Count`, the most useful statistic is `Average`.

Dimensions for Amazon EC2 Spot Fleet Metrics

You can filter the data using the following dimensions.

Dimensions	Description
`AvailabilityZone`	Filter the data by Availability Zone.

Dimensions	Description
FleetRequestId	Filter the data by Spot Fleet request.
InstanceType	Filter the data by instance type.

Amazon ECS Metrics and Dimensions

Amazon Elastic Container Service (Amazon ECS) sends metrics to Amazon CloudWatch. For more information, see Amazon ECS CloudWatch Metrics in the *Amazon Elastic Container Service Developer Guide*.

Amazon ECS Metrics

Amazon ECS provides metrics for you to monitor the CPU and memory reservation and utilization across your cluster as a whole, and the CPU and memory utilization on the services in your clusters.

The metrics made available will depend on the launch type of the tasks and services in your clusters. If you are using the Fargate launch type for your services then CPU and memory utilization metrics are provided to assist in the monitoring of your services. For the Amazon EC2 launch type you will own and need to monitor the EC2 instances that make your underlying infrastructure so additional CPU and memory reservation and utilization metrics are made available at the cluster, service, and task level.

Amazon ECS sends the following metrics to CloudWatch every minute. When Amazon ECS collects metrics, it collects multiple data points every minute. It then aggregates them to one data point before sending the data to CloudWatch. So in CloudWatch, one sample count is actually the aggregate of multiple data points during one minute.

The `AWS/ECS` namespace includes the following metrics.

Metric	Description
CPUReservation	The percentage of CPU units that are reserved by running tasks in the cluster.
	Cluster CPU reservation (this metric can only be filtered by `ClusterName`) is measured as the total CPU units that are reserved by Amazon ECS tasks on the cluster, divided by the total CPU units that were registered for all of the container instances in the cluster. This metric is only used for tasks using the EC2 launch type.
	Valid Dimensions: `ClusterName`
	Valid Statistics: Average, Minimum, Maximum, Sum, Sample Count. The most useful statistic is Average.
	Unit: Percent
CPUUtilization	The percentage of CPU units that are used in the cluster or service.
	Cluster CPU utilization (metrics that are filtered by `ClusterName` without `ServiceName`) is measured as the total CPU units in use by Amazon ECS tasks on the cluster, divided by the total CPU units that were registered for all of the container instances in the

Metric	Description
	cluster. Cluster CPU utilization metrics are only used for tasks using the EC2 launch type.
	Service CPU utilization (metrics that are filtered by `ClusterName` and `ServiceName`) is measured as the total CPU units in use by the tasks that belong to the service, divided by the total number of CPU units that are reserved for the tasks that belong to the service. Service CPU utilization metrics are used for tasks using both the Fargate and the EC2 launch type.
	Valid Dimensions: `ClusterName`, `ServiceName`
	Valid Statistics: Average, Minimum, Maximum, Sum, Sample Count. The most useful statistic is Average.
	Unit: Percent
`MemoryReservation`	The percentage of memory that is reserved by running tasks in the cluster.
	Cluster memory reservation (this metric can only be filtered by `ClusterName`) is measured as the total memory that is reserved by Amazon ECS tasks on the cluster, divided by the total amount of memory that was registered for all of the container instances in the cluster. This metric is only used for tasks using the EC2 launch type.
	Valid Dimensions: `ClusterName`
	Valid Statistics: Average, Minimum, Maximum, Sum, Sample Count. The most useful statistic is Average.
	Unit: Percent

Metric	Description
MemoryUtilization	The percentage of memory that is used in the cluster or service. Cluster memory utilization (metrics that are filtered by `ClusterName` without `ServiceName`) is measured as the total memory in use by Amazon ECS tasks on the cluster, divided by the total amount of memory that was registered for all of the container instances in the cluster. Cluster memory utilization metrics are only used for tasks using the EC2 launch type. Service memory utilization (metrics that are filtered by `ClusterName` and `ServiceName`) is measured as the total memory in use by the tasks that belong to the service, divided by the total memory that is reserved for the tasks that belong to the service. Service memory utilization metrics are used for tasks using both the Fargate and the EC2 launch type. Valid Dimensions: `ClusterName`, `ServiceName` Valid Statistics: Average, Minimum, Maximum, Sum, Sample Count. The most useful statistic is Average. Unit: Percent

Note

If you are using tasks with the EC2 launch type and have Linux container instances, the Amazon ECS container agent relies on Docker `stats` metrics to gather CPU and memory data for each container running on the instance. If you are using an Amazon ECS agent prior to version 1.14.0, ECS includes filesystem cache usage when reporting memory utilization to CloudWatch so your CloudWatch graphs show a higher than actual memory utilization for tasks. To remediate this, starting with Amazon ECS agent version 1.14.0, the Amazon ECS container agent excludes the filesystem cache usage from the memory utilization metric. This change does not impact the out-of-memory behavior of containers.

Dimensions for Amazon ECS Metrics

Amazon ECS metrics use the `AWS/ECS` namespace and provide metrics for the following dimensions:

Dimension	Description
ClusterName	This dimension filters the data you request for all resources in a specified cluster. All Amazon ECS metrics are filtered by `ClusterName`.
ServiceName	This dimension filters the data you request for all resources in a specified service within a specified cluster.

AWS Elastic Beanstalk Metrics and Dimensions

AWS Elastic Beanstalk sends metrics to Amazon CloudWatch. For more information, see Publishing Amazon CloudWatch Custom Metrics for an Environment in the *AWS Elastic Beanstalk Developer Guide*.

Elastic Beanstalk Metrics

The `AWS/ElasticBeanstalk` namespace includes the following metrics.

Metric	Description
`EnvironmentHealth`	[Environment] The health status of the environment. The possible values are 0 (OK), 1 (Info), 5 (Unknown), 10 (No data), 15 (Warning), 20 (Degraded) and 25 (Severe).
`InstancesOk`	[Environment] The number of instances with OK health status.
`InstancesPending`	[Environment] The number of instances with Pending health status.
`InstancesInfo`	[Environment] The number of instances with Info health status.
`InstancesUnknown`	[Environment] The number of instances with Unknown health status.
`InstancesNoData`	[Environment] The number of instances with no health status data.
`InstancesWarning`	[Environment] The number of instances with Warning health status.
`InstancesDegraded`	[Environment] The number of instances with Degraded health status.
`InstancesSevere`	[Environment] The number of instances with Severe health status.
`ApplicationRequestsTotal`	The number of requests completed by the instance or environment.
`ApplicationRequests2xx`	The number of requests that completed with a 2XX status code.
`ApplicationRequests3xx`	The number of requests that completed with a 3XX status code.
`ApplicationRequests4xx`	The number of requests that completed with a 4XX status code.
`ApplicationRequests5xx`	The number of requests that completed with a 5XX status code.
`ApplicationLatencyP10`	The average time to complete the fastest 10 percent of requests.
`ApplicationLatencyP50`	The average time to complete the fastest 50 percent of requests.
`ApplicationLatencyP75`	The average time to complete the fastest 75 percent of requests.
`ApplicationLatencyP85`	The average time to complete the fastest 85 percent of requests.
`ApplicationLatencyP90`	The average time to complete the fastest 90 percent of requests.
`ApplicationLatencyP95`	The average time to complete the fastest 95 percent of requests.
`ApplicationLatencyP99`	The average time to complete the fastest 99 percent of requests.
`ApplicationLatencyP99.9`	The average time to complete the fastest x percent of requests.
`LoadAverage1min`	[Instance] The average CPU load over the last minute.

Metric	Description
InstanceHealth	[Instance] The health status of the instance.
RootFilesystemUtil	[Instance] The percentage of disk space in use.
CPUIrq	[Instance] The percentage of time the CPU was in this state in the last minute.
CPUUser	[Instance] The percentage of time the CPU was in this state in the last minute.
CPUIdle	[Instance] The percentage of time the CPU was in this state in the last minute.
CPUSystem	[Instance] The percentage of time the CPU was in this state in the last minute.
CPUSoftirq	[Instance] The percentage of time the CPU was in this state in the last minute.
CPUIowait	[Instance] The percentage of time the CPU was in this state in the last minute.
CPUNice	[Instance] The percentage of time the CPU was in this state in the last minute.

Dimensions for Elastic Beanstalk Metrics

You can filter the data using the following dimensions.

Dimensions	Description
EnvironmentName	Filter the data by environment.
InstanceId	Filter the data by instance.

Amazon ElastiCache Metrics and Dimensions

Amazon ElastiCache sends metrics to Amazon CloudWatch. For more information, see Viewing Cache Cluster and Cache Node Metrics in the *Amazon ElastiCache User Guide*.

Contents

Dimensions for ElastiCache Metrics

All ElastiCache metrics use the AWS/ElastiCache namespace and provide metrics for a single dimension, the CacheNodeId, which is the automatically-generated identifier for each cache node

in the cache cluster. You can find out what these values are for your cache nodes by using the `DescribeCacheClusters` API or **describe-cache-clusters** command line utility. For more information, see DescribeCacheClusters in the *Amazon ElastiCache API Reference* and describe-cache-clusters in the *AWS CLI Command Reference*.

Each metric is published under a single set of dimensions. When retrieving metrics, you must supply both the `CacheClusterId` and `CacheNodeId` dimensions.

Contents

Host-Level Metrics

The `AWS/ElastiCache` namespace includes the following host-level metrics for individual cache nodes.

See Also

Metric	Description	Unit
`CPUUtilization`	The percentage of CPU utilization for the entire host. Because Redis is single-threaded, we recommend you monitor `EngineCPUUtilization` metric if available.	Percent
`FreeableMemory`	The amount of free memory available on the host. This is derived from the RAM, buffers and cache that the OS reports as freeable.	Bytes
`NetworkBytesIn`	The number of bytes the host has read from the network.	Bytes
`NetworkBytesOut`	The number of bytes the host has written to the network.	Bytes
`SwapUsage`	The amount of swap used on the host.	Bytes

Metrics for Memcached

The `AWS/ElastiCache` namespace includes the following metrics that are derived from the Memcached **stats** command. Each metric is calculated at the cache node level.

For complete documentation of the Memcached **stats** command, go to https://github.com/memcached/memcached/blob/master/doc/protocol.txt.

See Also

Metric	Description	Unit
`BytesReadIntoMemcached`	The number of bytes that have been read from the network by the cache node.	Bytes
`BytesUsedForCacheItems`	The number of bytes used to store cache items.	Bytes
`BytesWrittenOutFromMemcached`	The number of bytes that have been written to the network by the cache node.	Bytes
`CasBadval`	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
`CasHits`	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
`CasMisses`	The number of Cas requests the cache has received where the key requested was not found.	Count
`CmdFlush`	The number of **flush** commands the cache has received.	Count
`CmdGet`	The number of **get** commands the cache has received.	Count
`CmdSet`	The number of **set** commands the cache has received.	Count
`CurrConnections`	A count of the number of connections connected to the cache at an instant in time. ElastiCache uses two to three of the connections to monitor the cluster in each case.	Count
`CurrItems`	A count of the number of items currently stored in the cache.	Count
`DecrHits`	The number of decrement requests the cache has received where the requested key was found.	Count
`DecrMisses`	The number of decrement requests the cache has received where the requested key was not found.	Count
`DeleteHits`	The number of delete requests the cache has received where the requested key was found.	Count
`DeleteMisses`	The number of delete requests the cache has received where the requested key was not found.	Count
`Evictions`	The number of non-expired items the cache evicted to allow space for new writes.	Count
`GetHits`	The number of get requests the cache has received where the key requested was found.	Count
`GetMisses`	The number of get requests the cache has received where the key requested was not found.	Count

Metric	Description	Unit
`IncrHits`	The number of increment requests the cache has received where the key requested was found.	Count
`IncrMisses`	The number of increment requests the cache has received where the key requested was not found.	Count
`Reclaimed`	The number of expired items the cache evicted to allow space for new writes.	Count

For Memcached 1.4.14, the following additional metrics are provided.

Metric	Description	Unit
`BytesUsedForHash`	The number of bytes currently used by hash tables.	Bytes
`CmdConfigGet`	The cumulative number of **config get** requests.	Count
`CmdConfigSet`	The cumulative number of **config set** requests.	Count
`CmdTouch`	The cumulative number of **touch** requests.	Count
`CurrConfig`	The current number of configurations stored.	Count
`EvictedUnfetched`	The number of valid items evicted from the least recently used cache (LRU) which were never touched after being set.	Count
`ExpiredUnfetched`	The number of expired items reclaimed from the LRU which were never touched after being set.	Count
`SlabsMoved`	The total number of slab pages that have been moved.	Count
`TouchHits`	The number of keys that have been touched and were given a new expiration time.	Count
`TouchMisses`	The number of items that have been touched, but were not found.	Count

The `AWS/ElastiCache` namespace includes the following calculated cache-level metrics.

Metric	Description	Unit
`NewConnections`	The number of new connections the cache has received. This is derived from the memcached `total_connections` statistic by recording the change in `total_connections` across a period of time. This will always be at least 1, due to a connection reserved for a ElastiCache.	Count
`NewItems`	The number of new items the cache has stored. This is derived from the memcached `total_items` statistic by recording the change in `total_items` across a period of time.	Count

Metric	Description	Unit
UnusedMemory	The amount of memory not used by data. This is derived from the Memcached statistics `limit_maxbytes` and `bytes` by subtracting `bytes` from `limit_maxbytes`. Because Memcached overhead uses memory in addition to that used by data, `UnusedMemory` should not be considered to be the amount of memory available for additional data. You may experience evictions even though you still have some unused memory. For more detailed information, see Memcached item memory usage.	Bytes

Metrics for Redis

The `AWS/ElastiCache` namespace includes the following Redis metrics.

With the exception of `ReplicationLag` and `EngineCPUUtilization`, these metrics are derived from the Redis **info** command. Each metric is calculated at the cache node level.

For complete documentation of the Redis **info** command, see http://redis.io/commands/info.

See Also

-

Metric	Description	Unit
ActiveDefragHits	The number of value reallocations per minute performed by the active defragmentation process. This is derived from `active_defrag_hits` statistic at Redis INFO.	Number
BytesUsedForCache	The total number of bytes allocated by Redis for all purposes, including the dataset, buffers, etc. This is derived from `used_memory` statistic at Redis INFO.	Bytes
CacheHits	The number of successful read-only key lookups in the main dictionary. This is derived from `keyspace_hits` statistic at Redis INFO.	Count
CacheMisses	The number of unsuccessful read-only key lookups in the main dictionary. This is derived from `keyspace_misses` at Redis INFO.	Count
CurrConnections	The number of client connections, excluding connections from read replicas. ElastiCache uses two to three of the connections to monitor the cluster in each case. This is derived from the `connected_clients` statistic at Redis INFO.	Count

Metric	Description	Unit
EngineCPUUtilization	`EngineCPUUtilization` provides access to the Redis process CPU utilization to gain better insights into your Redis workloads. As Redis is single threaded and uses just one CPU core at any given point in time, `EngineCPUUtilization` provides more precise visibility into the load of the Redis process itself. `EngineCPUUtilization` adds to the pre-existing `CPUUtilization` metric which exposes CPU utilization for the server instance as a whole including other operating system and management processes. We recommend that you use both `EngineCPUUtilization` and `CPUUtilization` metrics together to get a detailed understanding of CPU utilization for your Redis environment.	Percent
Evictions	The number of keys that have been evicted due to the `maxmemory` limit. This is derived from the `evicted_keys` statistic at Redis INFO.	Count
NewConnections	The total number of connections that have been accepted by the server during this period. This is derived from the `total_connections_received` statistic at Redis INFO.	Count
Reclaimed	The total number of key expiration events. This is derived from the `expired_keys` statistic at Redis INFO.	Count
ReplicationBytes	For nodes in a replicated configuration, ReplicationBytes reports the number of bytes that the primary is sending to all of its replicas. This metric is representative of the write load on the replication group. This is derived from the `master_repl_offset` statistic at Redis INFO.	Bytes
ReplicationLag	This metric is only applicable for a node running as a read replica. It represents how far behind, in seconds, the replica is in applying changes from the primary node.	Seconds
SaveInProgress	This binary metric returns 1 whenever a background save (forked or forkless) is in progress, and 0 otherwise. A background save process is typically used during snapshots and syncs. These operations can cause degraded performance. Using the `SaveInProgress` metric, you can diagnose whether or not degraded performance was caused by a background save process. This is derived from the `rdb_bgsave_in_progress` statistic at Redis INFO.	Count

EngineCPUUtilization availability

Nodes in a region created or replaced after the date and time specified in the following table will include the `EngineCPUUtilization` metric.

Region	Region name	EngineCPUUtilization availability	
`us-east-2`	US East (Ohio)	February 16, 2017	17:21 (UTC)
`us-east-1`	US East (N. Virginia)	February 8, 2017	21:20 (UTC)
`us-west-1`	US West (N. California)	February 14, 2017	22:23 (UTC)
`us-west-2`	US West (Oregon)	Febrary 20, 2017	19:20 (UTC)
`ap-northeast-1`	Asia Pacific (Tokyo)	February 14, 2017	19:58 (UTC)
`ap-northeast-2`	Asia Pacific (Seoul)	Available on all nodes.	
`ap-northeast-3`	Asia Pacific (Osaka-Local)	Available on all nodes.	
`ap-south-1`	Asia Pacific (Mumbai)	February 7, 2017	02:51 (UTC)
`ap-southeast-1`	Asia Pacific (Singapore)	February 13, 2017	23:40 (UTC)
`ap-southeast-2`	Asia Pacific (Sydney)	February 14, 2017	03:33 (UTC)
`ca-central-1`	Canada (Central)	Available on all nodes.	
`cn-north-1`	China (Beijing)	February 16, 2017	22:39 (UTC)
`cn-northwest-2`	China (Ningxia)	Available on all nodes.	
`eu-central-1`	EU (Frankfurt)	February 15, 2017	00:46 (UTC)
`eu-west-1`	EU (Ireland)	February 7, 2017	21:30 (UTC)
`eu-west-2`	EU (London)	February 16, 2017	18:58 (UTC)
`eu-west-3`	EU (Paris)	Available on all nodes.	
`sa-east-1`	South America (São Paulo)	February 7, 2017	04:35 (UTC)
`us-gov-west-1`	AWS GovCloud (US)	February 16, 2017	20:11 (UTC)

These are aggregations of certain kinds of commands, derived from **info commandstats**:

Metric	Description	Unit
`CurrItems`	The number of items in the cache. This is derived from the Redis `keyspace` statistic, summing all of the keys in the entire keyspace.	Count
`GetTypeCmds`	The total number of **read-only** type commands. This is derived from the Redis `commandstats` statistic by summing all of the **read-only** type commands (**get**, **hget**, **scard**, **lrange**, etc.)	Count
`HashBasedCmds`	The total number of commands that are hash-based. This is derived from the Redis	Count

Metric	Description	Unit
	`commandstats` statistic by summing all of the commands that act upon one or more hashes.	
`HyperLogLogBasedCmds`	The total number of `HyperLogLog`-based commands. This is derived from the Redis `commandstats` statistic by summing all of the **pf** type of commands (**pfadd**, **pfcount**, **pfmerge**, etc.).	Count
`KeyBasedCmds`	The total number of commands that are key-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more keys across multiple data structures (**del**, **expire**, **rename**, etc.).	Count
`ListBasedCmds`	The total number of commands that are list-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more lists.	Count
`SetBasedCmds`	The total number of commands that are set-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more sets.	Count
`SetTypeCmds`	The total number of **write** types of commands. This is derived from the Redis `commandstats` statistic by summing all of the **mutative** types of commands that operate on data (**set**, **hset**, **sadd**, **lpop**, etc.)	Count
`SortedSetBasedCmds`	The total number of commands that are sorted set-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more sorted sets.	Count
`StringBasedCmds`	The total number of commands that are string-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more strings.	Count

Amazon EBS Metrics and Dimensions

Amazon EBS Metrics

Amazon Elastic Block Store (Amazon EBS) sends data points to CloudWatch for several metrics. Amazon EBS General Purpose SSD (gp2), Throughput Optimized HDD (st1) , Cold HDD (sc1), and Magnetic (standard) volumes automatically send five-minute metrics to CloudWatch. Provisioned IOPS SSD (io1) volumes automatically send one-minute metrics to CloudWatch. Data is only reported to CloudWatch when the volume is attached to an instance. For more information about how to monitor Amazon EBS, see Monitoring the Status of Your Volumes in the *Amazon EC2 User Guide for Linux Instances*.

The `AWS/EBS` namespace includes the following metrics.

Metric	Description
`VolumeReadBytes`	Provides information on the read operations in a specified period of time. The `Sum` statistic reports the total number of bytes transferred during the period. The `Average` statistic reports the average size of each read operation during the period, except on volumes attached to a Nitro instance, where the average represents the average over the specified period. The `SampleCount` statistic reports the total number of read operations during the period, except on volumes attached to a Nitro instance, where the sample count represents the number of data points used in the statistical calculation. For Xen instances, Cloudwatch data is reported only when there is read activity on the attached EBS volume. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to a C5 instance. Units: Bytes
`VolumeWriteBytes`	Provides information on the write operations in a specified period of time. The `Sum` statistic reports the total number of bytes transferred during the period. The `Average` statistic reports the average size of each write operation during the period, except on volumes attached to a Nitro instance, where the average represents the average over the specified period. The `SampleCount` statistic reports the total number of write operations during the period, except on volumes attached to a Nitro instance, where the sample count represents the number of data points used in the statistical calculation. For Xen instances, Cloudwatch data is reported only when there is write activity on the attached EBS volume. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to the following instances: . Units: Bytes
`VolumeReadOps`	The total number of read operations in a specified period of time. To calculate the average read operations per second (read IOPS) for the period, divide the total read operations in the period by the number of seconds in that period. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to a C5 instance. Units: Count
`VolumeWriteOps`	The total number of write operations in a specified period of time. To calculate the average write operations per second (write IOPS) for the period, divide the total write operations in the period by the number of seconds in that period. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to the following instances: .

Metric	Description
	Units: Count
`VolumeTotalReadTime`	The total number of seconds spent by all read operations that completed in a specified period of time. If multiple requests are submitted at the same time, this total could be greater than the length of the period. For example, for a period of 5 minutes (300 seconds): if 700 operations completed during that period, and each operation took 1 second, the value would be 700 seconds. For Xen instances, Cloudwatch data is reported only when there is read activity on the attached EBS volume. The `Average` statistic on this metric is not relevant for volumes attached to C5 instances. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to a C5 instance. Units: Seconds
`VolumeTotalWriteTime`	The total number of seconds spent by all write operations that completed in a specified period of time. If multiple requests are submitted at the same time, this total could be greater than the length of the period. For example, for a period of 5 minutes (300 seconds): if 700 operations completed during that period, and each operation took 1 second, the value would be 700 seconds. For Xen instances, Cloudwatch data is reported only when there is write activity on the attached EBS volume. The `Average` statistic on this metric is not relevant for volumes attached to the following instances: . The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to the following instances: . Units: Seconds
`VolumeIdleTime`	The total number of seconds in a specified period of time when no read or write operations were submitted. The `Average` statistic on this metric is not relevant for volumes attached to C5 instances. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to a C5 instance. Units: Seconds
`VolumeQueueLength`	The number of read and write operation requests waiting to be completed in a specified period of time. The `Sum` statistic on this metric is not relevant for volumes attached to C5 instances. The `Minimum` and `Maximum` statistics on this metric are supported only by volumes attached to a C5 instance. Units: Count

Metric	Description
`VolumeThroughputPercentage`	Used with Provisioned IOPS SD volumes only. The percentage of I/O operations per second (IOPS) delivered of the total IOPS provisioned for an Amazon EBS volume. Provisioned IOPS SD volumes deliver within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year. During a write, if there are no other pending I/O requests in a minute, the metric value will be 100 percent. Also, a volume's I/O performance may become degraded temporarily due to an action you have taken (for example, creating a snapshot of a volume during peak usage, running the volume on a non-EBS-optimized instance, or accessing data on the volume for the first time). Units: Percent
`VolumeConsumedReadWriteOps`	Used with Provisioned IOPS SD volumes only. The total amount of read and write operations (normalized to 256K capacity units) consumed in a specified period of time. I/O operations that are smaller than 256K each count as 1 consumed IOPS. I/O operations that are larger than 256K are counted in 256K capacity units. For example, a 1024K I/O would count as 4 consumed IOPS. Units: Count
`BurstBalance`	Used with General Purpose SSD (`gp2`), Throughput Optimized HDD (`st1`), and Cold HDD (`sc1`) volumes only. Provides information about the percentage of I/O credits (for `gp2`) or throughput credits (for `st1` and `sc1`) remaining in the burst bucket. Data is reported to CloudWatch only when the volume is active. If the volume is not attached, no data is reported. The `Sum` statistic on this metric is not relevant for volumes attached to C5 instances. Units: Percent

Dimensions for Amazon EBS Metrics

Dimensions for Amazon EBS Metrics

The only dimension that Amazon EBS sends to CloudWatch is the volume ID. This means that all available statistics are filtered by volume ID.

Amazon EFS Metrics and Dimensions

Amazon EFS sends metrics to CloudWatch for every Amazon EFS file system every minute. For more information, see Monitoring Amazon EFS in the *Amazon Elastic File System User Guide*.

Amazon CloudWatch Metrics for Amazon EFS

The `AWS/EFS` namespace includes the following metrics.

Metric	Description
BurstCreditBalance	The number of burst credits that a file system has.
	Burst credits allow a file system to burst to throughput levels above a file system's baseline level for periods of time. For more information, see Throughput scaling in Amazon EFS.
	The `Minimum` statistic is the smallest burst credit balance for any minute during the period. The `Maximum` statistic is the largest burst credit balance for any minute during the period. The `Average` statistic is the average burst credit balance during the period.
	Units: Bytes
	Valid statistics: `Minimum`, `Maximum`, `Average`
ClientConnections	The number of client connections to a file system. When using a standard client, there is one connection per mounted Amazon EC2 instance.
	Note To calculate the average `ClientConnections` for periods greater than one minute, divide the `Sum` statistic by the number of minutes in the period.
	Units: Count of client connections
	Valid statistics: `Sum`
DataReadIOBytes	The number of bytes for each file system read operation.
	The `Sum` statistic is the total number of bytes associated with read operations. The `Minimum` statistic is the size of the smallest read operation during the period. The `Maximum` statistic is the size of the largest read operation during the period. The `Average` statistic is the average size of read operations during the period. The `SampleCount` statistic provides a count of read operations.
	Units:
	• Bytes for `Minimum`, `Maximum`, `Average`, and `Sum`. • Count for `SampleCount`.
	Valid statistics: `Minimum`, `Maximum`, `Average`, `Sum`, `SampleCount`
DataWriteIOBytes	The number of bytes for each file write operation.
	The `Sum` statistic is the total number of bytes associated with write operations. The `Minimum` statistic is the size of the smallest write operation during the period. The `Maximum` statistic is the size of the largest write operation during the period. The `Average` statistic is the average size of write operations during the period. The `SampleCount` statistic provides a count of write operations.

Metric	Description
	Units: • Bytes are the units for the `Minimum`, `Maximum`, `Average`, and `Sum` statistics. • Count for `SampleCount`. Valid statistics: `Minimum, Maximum, Average, Sum, SampleCount`
`MetadataIOBytes`	The number of bytes for each metadata operation. The `Sum` statistic is the total number of bytes associated with metadata operations. The `Minimum` statistic is the size of the smallest metadata operation during the period. The `Maximum` statistic is the size of the largest metadata operation during the period. The `Average` statistic is the size of the average metadata operation during the period. The `SampleCount` statistic provides a count of metadata operations. Units: • Bytes are the units for the `Minimum`, `Maximum`, `Average`, and `Sum` statistics. • Count for `SampleCount`. Valid statistics: `Minimum, Maximum, Average, Sum, SampleCount`
`PercentIOLimit`	Shows how close a file system is to reaching the I/O limit of the General Purpose performance mode. If this metric is at 100% more often than not, consider moving your application to a file system using the Max I/O performance mode. **Note** This metric is only submitted for file systems using the General Purpose performance mode. Units: • Percent
`PermittedThroughput`	The maximum amount of throughput a file system is allowed. For file systems in the Provisioned Throughput mode, this value is the same as the provisioned throughput. For file systems in the Bursting Throughput mode, this value is a function of the file system size and `BurstCreditBalance`. For more information, see Amazon EFS Performance. The `Minimum` statistic is the smallest throughput permitted for any minute during the period. The `Maximum` statistic is the highest throughput permitted for any minute during the period. The `Average` statistic is the average throughput permitted during the period. Units: Bytes per second Valid statistics: `Minimum, Maximum, Average`

Metric	Description
TotalIOBytes	The number of bytes for each file system operation, including data read, data write, and metadata operations. The Sum statistic is the total number of bytes associated with all file system operations. The Minimum statistic is the size of the smallest operation during the period. The Maximum statistic is the size of the largest operation during the period. The Average statistic is the average size of an operation during the period. The SampleCount statistic provides a count of all operations. **Note** To calculate the average operations per second for a period, divide the SampleCount statistic by the number of seconds in the period. To calculate the average throughput (Bytes per second) for a period, divide the Sum statistic by the number of seconds in the period. Units: • Bytes for Minimum, Maximum, Average, and Sum statistics. • Count for SampleCount. Valid statistics: Minimum, Maximum, Average, Sum, SampleCount

Dimensions for Amazon EFS Metrics

Amazon EFS Dimensions

Amazon EFS metrics use the EFS namespace and provides metrics for a single dimension, FileSystemId. A file system's ID can be found in the Amazon EFS management console, and it takes the form of fs-XXXXXXXX.

Elastic Load Balancing Metrics and Dimensions

Elastic Load Balancing supports three types of load balancers: Classic Load Balancers, Application Load Balancers and Network Load Balancers. Elastic Load Balancing sends metrics to CloudWatch for all three types of load balancers.

Contents

Application Load Balancer Metrics

The AWS/ApplicationELB namespace includes the following metrics for load balancers.

Metric	Description
`ActiveConnectionCount`	The total number of concurrent TCP connections active from clients to the load balancer and from the load balancer to targets. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. **Dimensions** • `LoadBalancer`
`ClientTLSNegotiationErrorCount`	The number of TLS connections initiated by the client that did not establish a session with the load balancer. Possible causes include a mismatch of ciphers or protocols. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. **Dimensions** • `LoadBalancer` • `AvailabilityZone, LoadBalancer`
`ConsumedLCUs`	The number of load balancer capacity units (LCU) used by your load balancer. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing Pricing. **Reporting criteria**: Always reported **Statistics**: All **Dimensions** • `LoadBalancer`
`HTTP_Fixed_Response_Count`	The number of fixed-response actions that were successful. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is `Sum`. **Dimensions** • `LoadBalancer`
`HTTP_Redirect_Count`	The number of redirect actions that were successful. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is `Sum`. **Dimensions** • `LoadBalancer`
`HTTP_Redirect_Url_Limit_Exceeded_Count`	The number of redirect actions that couldn't be completed because the URL in the response location header is larger than 8K.

Metric	Description
	Reporting criteria: There is a nonzero value **Statistics**: The only meaningful statistic is `Sum`. **Dimensions** • `LoadBalancer`
`HTTPCode_ELB_3XX_Count`	The number of HTTP 3XX redirection codes that originate from the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is `Sum`. **Dimensions** • `LoadBalancer`
`HTTPCode_ELB_4XX_Count`	The number of HTTP 4XX client error codes that originate from the load balancer. Client errors are generated when requests are malformed or incomplete. These requests have not been received by the target. This count does not include any response codes generated by the targets. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1. **Dimensions** • `LoadBalancer` • `AvailabilityZone, LoadBalancer`
`HTTPCode_ELB_5XX_Count`	The number of HTTP 5XX server error codes that originate from the load balancer. This count does not include any response codes generated by the targets. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1. **Dimensions** • `LoadBalancer` • `AvailabilityZone, LoadBalancer`
`IPv6ProcessedBytes`	The total number of bytes processed by the load balancer over IPv6. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. **Dimensions** • `LoadBalancer`

Metric	Description
IPv6RequestCount	The number of IPv6 requests received by the load balancer.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.
	Dimensions
	`LoadBalancer` `AvailabilityZone, LoadBalancer` `TargetGroup, LoadBalancer` `TargetGroup, AvailabilityZone, LoadBalancer`
NewConnectionCount	The total number of new TCP connections established from clients to the load balancer and from the load balancer to targets.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`.
	Dimensions
	`LoadBalancer`
ProcessedBytes	The total number of bytes processed by the load balancer over IPv4 and IPv6.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`.
	Dimensions
	`LoadBalancer`
RejectedConnectionCount	The number of connections that were rejected because the load balancer had reached its maximum number of connections.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`.
	Dimensions
	`LoadBalancer` `AvailabilityZone, LoadBalancer`

Metric	Description
RequestCount	The number of requests processed over IPv4 and IPv6. This count includes only the requests with a response generated by a target of the load balancer. **Reporting criteria**: Always reported **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer • AvailabilityZone, LoadBalancer • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer
RuleEvaluations	The number of rules processed by the load balancer given a request rate averaged over an hour. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. **Dimensions** • LoadBalancer

The AWS/ApplicationELB namespace includes the following metrics for targets.

Metric	Description
HealthyHostCount	The number of targets that are considered healthy. **Reporting criteria**: Always reported **Statistics**: The most useful statistics are Average, Minimum, and Maximum. **Dimensions** • TargetGroup, LoadBalancer • TargetGroup, AvailabilityZone, LoadBalancer
HTTPCode_Target_2XX_Count, HTTPCode_Target_3XX_Count, HTTPCode_Target_4XX_Count, HTTPCode_Target_5XX_Count	The number of HTTP response codes generated by the targets. This does not include any response codes generated by the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. Note that Minimum, Maximum, and Average all return 1. **Dimensions** • LoadBalancer • AvailabilityZone, LoadBalancer

Metric	Description
	• `TargetGroup, LoadBalancer`
	• `TargetGroup, AvailabilityZone, LoadBalancer`
`RequestCountPerTarget`	The average number of requests received by each target in a target group. You must specify the target group using the `TargetGroup` dimension.
	Reporting criteria: Always reported
	Statistics: The only valid statistic is `Sum`. Note that this represents the average not the sum.
	Dimensions
	• `TargetGroup`
	• `TargetGroup, LoadBalancer`
`TargetConnectionErrorCount`	The number of connections that were not successfully established between the load balancer and target.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`.
	Dimensions
	• `LoadBalancer`
	• `AvailabilityZone, LoadBalancer`
	• `TargetGroup, LoadBalancer`
	• `TargetGroup, AvailabilityZone, LoadBalancer`
`TargetResponseTime`	The time elapsed, in seconds, after the request leaves the load balancer until a response from the target is received. This is equivalent to the `target_processing_time` field in the access logs.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistics are `Average` and `pNN.NN` (percentiles).
	Dimensions
	• `LoadBalancer`
	• `AvailabilityZone, LoadBalancer`
	• `TargetGroup, LoadBalancer`
	• `TargetGroup, AvailabilityZone, LoadBalancer`

Metric	Description
TargetTLSNegotiationErrorCount	The number of TLS connections initiated by the load balancer that did not establish a session with the target. Possible causes include a mismatch of ciphers or protocols. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. **Dimensions** • `LoadBalancer` • `AvailabilityZone, LoadBalancer` • `TargetGroup, LoadBalancer` • `TargetGroup, AvailabilityZone, LoadBalancer`
UnHealthyHostCount	The number of targets that are considered unhealthy. **Reporting criteria**: Always reported **Statistics**: The most useful statistics are `Average`, `Minimum`, and `Maximum`. **Dimensions** • `TargetGroup, LoadBalancer` • `TargetGroup, AvailabilityZone, LoadBalancer`

The `AWS/ApplicationELB` namespace includes the following metrics for user authentication.

Metric	Description
ELBAuthError	The number of user authentications that could not be completed because an authenticate action was misconfigured, the load balancer couldn't establish a connection with the IdP, or the load balancer couldn't complete the authentication flow due to an internal error. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is `Sum`. **Dimensions**: `LoadBalancer`
ELBAuthFailure	The number of user authentications that could not be completed because the IdP denied access to the user or an authorization code was used more than once. **Reporting criteria**: There is a nonzero value **Statistics**: The only meaningful statistic is `Sum`. **Dimensions**: `LoadBalancer`
ELBAuthLatency	The time elapsed, in milliseconds, to query the IdP for the ID token and user info. If one or more of these operations fail, this is the time to failure.

Metric	Description
	Reporting criteria: There is a nonzero value
	Statistics: All statistics are meaningful.
	Dimensions: `LoadBalancer`
`ELBAuthSuccess`	The number of authenticate actions that were successful. This metric is incremented at the end of the authentication workflow, after the load balancer has retrieved the user claims from the IdP.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`.
	Dimensions: `LoadBalancer`

Metric Dimensions for Application Load Balancers

To filter the metrics for your Application Load Balancer, use the following dimensions.

Dimension	Description
`AvailabilityZone`	Filter the metric data by Availability Zone.
`LoadBalancer`	Filter the metric data by load balancer. Specify the load balancer as follows: app/*load-balancer-name*/*1234567890123456* (the final portion of the load balancer ARN).
`TargetGroup`	Filter the metric data by target group. Specify the target group as follows: targetgroup/*target-group-name*/*1234567890123456* (the final portion of the target group ARN).

Network Load Balancer Metrics

The `AWS/NetworkELB` namespace includes the following metrics.

Metric	Description
`ActiveFlowCount`	The total number of concurrent TCP flows (or connections) from clients to targets. This metric includes connections in the SYN_SENT and ESTABLISHED states. TCP connections are not terminated at the load balancer, so a client opening a TCP connection to a target counts as a single flow.
	Statistics: The most useful statistics are `Average`, `Maximum`, and `Minimum`.
`ConsumedLCUs`	The number of load balancer capacity units (LCU) used by your load balancer. You pay for the number of LCUs that you use per hour. For more information, see Elastic Load Balancing Pricing.
`HealthyHostCount`	The number of targets that are considered healthy.

Metric	Description
	Statistics: The most useful statistics are `Maximum` and `Minimum`.
`NewFlowCount`	The total number of new TCP flows (or connections) established from clients to targets in the time period. **Statistics**: The most useful statistic is `Sum`.
`ProcessedBytes`	The total number of bytes processed by the load balancer, including TCP/IP headers. **Statistics**: The most useful statistic is `Sum`.
`TCP_Client_Reset_Count`	The total number of reset (RST) packets sent from a client to a target. These resets are generated by the client and forwarded by the load balancer. **Statistics**: The most useful statistic is `Sum`.
`TCP_ELB_Reset_Count`	The total number of reset (RST) packets generated by the load balancer. **Statistics**: The most useful statistic is `Sum`.
`TCP_Target_Reset_Count`	The total number of reset (RST) packets sent from a target to a client. These resets are generated by the target and forwarded by the load balancer. **Statistics**: The most useful statistic is `Sum`.
`UnHealthyHostCount`	The number of targets that are considered unhealthy. **Statistics**: The most useful statistics are `Maximum` and `Minimum`.

Metric Dimensions for Network Load Balancers

To filter the metrics for your load balancer, use the following dimensions.

Dimension	Description
`AvailabilityZone`	Filter the metric data by Availability Zone.
`LoadBalancer`	Filter the metric data by load balancer. Specify the load balancer as follows: net/*load-balancer-name*/*1234567890123456* (the final portion of the load balancer ARN).
`TargetGroup`	Filter the metric data by target group. Specify the target group as follows: targetgroup/*target-group-name*/*1234567890123456* (the final portion of the target group ARN).

Classic Load Balancer Metrics

The `AWS/ELB` namespace includes the following metrics.

Metric	Description
BackendConnectionErrors	The number of connections that were not successfully established between the load balancer and the registered instances. Because the load balancer retries the connection when there are errors, this count can exceed the request rate. Note that this count also includes any connection errors related to health checks. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. Note that Average, Minimum, and Maximum are reported per load balancer node and are not typically useful. However, the difference between the minimum and maximum (or peak to average or average to trough) might be useful to determine whether a load balancer node is an outlier. **Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that attempts to connect to 1 instance in us-west-2a result in back-end connection errors. The sum for us-west-2a includes these connection errors, while the sum for us-west-2b does not include them. Therefore, the sum for the load balancer equals the sum for us-west-2a.
HealthyHostCount	The number of healthy instances registered with your load balancer. A newly registered instance is considered healthy after it passes the first health check. If cross-zone load balancing is enabled, the number of healthy instances for the LoadBalancerName dimension is calculated across all Availability Zones. Otherwise, it is calculated per Availability Zone. **Reporting criteria**: There are registered instances **Statistics**: The most useful statistics are Average and Maximum. These statistics are determined by the load balancer nodes. Note that some load balancer nodes might determine that an instance is unhealthy for a brief period while other nodes determine that it is healthy. **Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, us-west-2a has 1 unhealthy instance, and us-west-2b has no unhealthy instances. With the AvailabilityZone dimension, there is an average of 1 healthy and 1 unhealthy instance in us-west-2a, and an average of 2 healthy and 0 unhealthy instances in us-west-2b.
HTTPCode_Backend_2XX, HTTPCode_Backend_3XX, HTTPCode_Backend_4XX, HTTPCode_Backend_5XX	[HTTP listener] The number of HTTP response codes generated by registered instances. This count does not include any response codes generated by the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is Sum. Note that Minimum, Maximum, and Average are all 1. **Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that requests sent to 1 instance in us-west-2a result in HTTP 500 responses. The sum for us-west-2a includes these error responses, while the sum for us-west-2b

Metric	Description
	does not include them. Therefore, the sum for the load balancer equals the sum for us-west-2a.
`HTTPCode_ELB_4XX`	[HTTP listener] The number of HTTP 4XX client error codes generated by the load balancer. Client errors are generated when a request is malformed or incomplete. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` are all 1. **Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that client requests include a malformed request URL. As a result, client errors would likely increase in all Availability Zones. The sum for the load balancer is the sum of the values for the Availability Zones.
`HTTPCode_ELB_5XX`	[HTTP listener] The number of HTTP 5XX server error codes generated by the load balancer. This count does not include any response codes generated by the registered instances. The metric is reported if there are no healthy instances registered to the load balancer, or if the request rate exceeds the capacity of the instances (spillover) or the load balancer. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` are all 1. **Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the surge queue for the load balancer nodes in us-west-2a fills and clients receive a 503 error. If us-west-2b continues to respond normally, the sum for the load balancer equals the sum for us-west-2a.
`Latency`	[HTTP listener] The total time elapsed, in seconds, from the time the load balancer sent the request to a registered instance until the instance started to send the response headers. [TCP listener] The total time elapsed, in seconds, for the load balancer to successfully establish a connection to a registered instance. **Reporting criteria**: There is a nonzero value **Statistics**: The most useful statistic is `Average`. Use `Maximum` to determine whether some requests are taking substantially longer than the average. Note that `Minimum` is typically not useful. **Example**: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that requests sent to 1 instance in us-west-2a have a higher latency. The average for us-west-2a has a higher value than the average for us-west-2b.

Metric	Description
`RequestCount`	The number of requests completed or connections made during the specified interval (1 or 5 minutes).
	[HTTP listener] The number of requests received and routed, including HTTP error responses from the registered instances.
	[TCP listener] The number of connections made to the registered instances.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`. Note that `Minimum`, `Maximum`, and `Average` all return 1.
	Example: Suppose that your load balancer has 2 instances in us-west-2a and 2 instances in us-west-2b, and that 100 requests are sent to the load balancer. There are 60 requests sent to us-west-2a, with each instance receiving 30 requests, and 40 requests sent to us-west-2b, with each instance receiving 20 requests. With the `AvailabilityZone` dimension, there is a sum of 60 requests in us-west-2a and 40 requests in us-west-2b. With the `LoadBalancerName` dimension, there is a sum of 100 requests.
`SpilloverCount`	The total number of requests that were rejected because the surge queue is full.
	[HTTP listener] The load balancer returns an HTTP 503 error code.
	[TCP listener] The load balancer closes the connection.
	Reporting criteria: There is a nonzero value
	Statistics: The most useful statistic is `Sum`. Note that `Average`, `Minimum`, and `Maximum` are reported per load balancer node and are not typically useful.
	Example: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the surge queue for the load balancer node in us-west-2a fills, resulting in spillover. If us-west-2b continues to respond normally, the sum for the load balancer will be the same as the sum for us-west-2a.

Metric	Description
SurgeQueueLength	The total number of requests (HTTP listener) or connections (TCP listener) that are pending routing to a healthy instance. The maximum size of the queue is 1,024. Additional requests or connections are rejected when the queue is full. For more information, see `SpilloverCount`. **Reporting criteria**: There is a nonzero value. **Statistics**: The most useful statistic is `Maximum`, because it represents the peak of queued requests. The `Average` statistic can be useful in combination with `Minimum` and `Maximum` to determine the range of queued requests. Note that `Sum` is not useful. **Example**: Suppose that your load balancer has us-west-2a and us-west-2b enabled, and that instances in us-west-2a are experiencing high latency and are slow to respond to requests. As a result, the surge queue for the load balancer nodes in us-west-2a fills, with clients likely experiencing increased response times. If this continues, the load balancer will likely have spillovers (see the `SpilloverCount` metric). If us-west-2b continues to respond normally, the `max` for the load balancer will be the same as the `max` for us-west-2a.
UnHealthyHostCount	The number of unhealthy instances registered with your load balancer. An instance is considered unhealthy after it exceeds the unhealthy threshold configured for health checks. An unhealthy instance is considered healthy again after it meets the healthy threshold configured for health checks. **Reporting criteria**: There are registered instances **Statistics**: The most useful statistics are `Average` and `Minimum`. These statistics are determined by the load balancer nodes. Note that some load balancer nodes might determine that an instance is unhealthy for a brief period while other nodes determine that it is healthy. **Example**: See `HealthyHostCount`.

The following metrics enable you to estimate your costs if you migrate a Classic Load Balancer to an Application Load Balancer. These metrics are intended for informational use only, not for use with CloudWatch alarms. Note that if your Classic Load Balancer has multiple listeners, these metrics are aggregated across the listeners.

These estimates are based on a load balancer with one default rule and a certificate that is 2K in size. If you use a certificate that is 4K or greater in size, we recommend that you estimate your costs as follows: create an Application Load Balancer based on your Classic Load Balancer using the migration tool and monitor the `ConsumedLCUs` metric for the Application Load Balancer. For more information, see Migrate from a Classic Load Balancer to an Application Load Balancer in the *Elastic Load Balancing User Guide*.

Metric	Description
EstimatedALBActiveConnectionCount	The estimated number of concurrent TCP connections active from clients to the load balancer and from the load balancer to targets.
EstimatedALBConsumedLCUs	The estimated number of load balancer capacity units (LCU) used by an Application Load Balancer. You pay for the number of LCUs that

Metric	Description
	you use per hour. For more information, see Elastic Load Balancing Pricing.
`EstimatedALBNewConnectionCount`	The estimated number of new TCP connections established from clients to the load balancer and from the load balancer to targets.
`EstimatedProcessedBytes`	The estimated number of bytes processed by an Application Load Balancer.

Metric Dimensions for Classic Load Balancers

To filter the metrics for your Classic Load Balancer, use the following dimensions.

Dimension	Description
`AvailabilityZone`	Filter the metric data by the specified Availability Zone.
`LoadBalancerName`	Filter the metric data by the specified load balancer.

Amazon EMR Metrics and Dimensions

Amazon EMR (Amazon EMR) sends metrics to CloudWatch. All Amazon EMR job flows automatically send metrics in five-minute intervals. Metrics are archived for 15 months; after that period, the data is discarded. For more information, see Monitor Metrics with Amazon CloudWatch in the *Amazon EMR Developer Guide*.

Amazon EMR Metrics

Amazon EMR sends the following metrics to Amazon CloudWatch.

The `AWS/ElasticMapReduce` namespace includes the following metrics.

Note
Amazon EMR pulls metrics from a cluster. If a cluster becomes unreachable, no metrics are reported until the cluster becomes available again.

The following are Hadoop 1 metrics:

Metric	Description
Cluster Status	
IsIdle	Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer. Use case: Monitor cluster performance

Metric	Description
	Units: *Boolean*
JobsRunning	The number of jobs in the cluster that are currently running.
	Use case: Monitor cluster health
	Units: *Count*
JobsFailed	The number of jobs in the cluster that have failed.
	Use case: Monitor cluster health
	Units: *Count*
Map/Reduce	
MapTasksRunning	The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.
	Use case: Monitor cluster progress
	Units: *Count*
MapTasksRemaining	The number of remaining map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. A remaining map task is one that is not in any of the following states: Running, Killed, or Completed.
	Use case: Monitor cluster progress
	Units: *Count*
MapSlotsOpen	The unused map task capacity. This is calculated as the maximum number of map tasks for a given cluster, less the total number of map tasks currently running in that cluster.
	Use case: Analyze cluster performance
	Units: *Count*
RemainingMapTasksPerSlot	The ratio of the total map tasks remaining to the total map slots available in the cluster.
	Use case: Analyze cluster performance
	Units: *Ratio*
ReduceTasksRunning	The number of running reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.
	Use case: Monitor cluster progress
	Units: *Count*

Metric	Description
ReduceTasksRemaining	The number of remaining reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. Use case: Monitor cluster progress Units: *Count*
ReduceSlotsOpen	Unused reduce task capacity. This is calculated as the maximum reduce task capacity for a given cluster, less the number of reduce tasks currently running in that cluster. Use case: Analyze cluster performance Units: *Count*
Node Status	
CoreNodesRunning	The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
CoreNodesPending	The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
LiveDataNodes	The percentage of data nodes that are receiving work from Hadoop. Use case: Monitor cluster health Units: *Percent*
TaskNodesRunning	The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
TaskNodesPending	The number of core nodes waiting to be assigned. All of the task nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*

Metric	Description
LiveTaskTrackers	The percentage of task trackers that are functional.
	Use case: Monitor cluster health
	Units: *Percent*
IO	
S3BytesWritten	The number of bytes written to Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR.
	Use case: Analyze cluster performance, Monitor cluster progress
	Units: *Count*
S3BytesRead	The number of bytes read from Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR.
	Use case: Analyze cluster performance, Monitor cluster progress
	Units: *Count*
HDFSUtilization	The percentage of HDFS storage currently used.
	Use case: Analyze cluster performance
	Units: *Percent*
HDFSBytesRead	The number of bytes read from HDFS.
	Use case: Analyze cluster performance, Monitor cluster progress
	Units: *Bytes*
HDFSBytesWritten	The number of bytes written to HDFS.
	Use case: Analyze cluster performance, Monitor cluster progress
	Units: *Bytes*
MissingBlocks	The number of blocks in which HDFS has no replicas. These might be corrupt blocks.
	Use case: Monitor cluster health
	Units: *Count*

Metric	Description
TotalLoad	The current, total number of readers and writers reported by all DataNodes in a cluster. Use case: Diagnose the degree to which high I/O might be contributing to poor job execution performance. Worker nodes running the DataNode daemon must also perform map and reduce tasks. Persistently high TotalLoad values over time can indicate that high I/O might be a contributing factor to poor performance. Occasional spikes in this value are typical and do not usually indicate a problem. Units: *Count*
HBase	
BackupFailed	Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: *Count*
MostRecentBackupDuration	The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters. Use case: Monitor HBase Backups Units: *Minutes*
TimeSinceLastSuccessfulBackup	The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: *Minutes*

The following metrics are available for Hadoop 2 AMIs:

Metric	Description
Cluster Status	
IsIdle	Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.

Metric	Description
	Use case: Monitor cluster performance Units: *Boolean*
ContainerAllocated	The number of resource containers allocated by the ResourceManager. Use case: Monitor cluster progress Units: *Count*
ContainerReserved	The number of containers reserved. Use case: Monitor cluster progress Units: *Count*
ContainerPending	The number of containers in the queue that have not yet been allocated. Use case: Monitor cluster progress Units: *Count*
ContainerPendingRatio	The ratio of pending containers to containers allocated (ContainerPendingRatio = ContainerPending / ContainerAllocated). If ContainerAllocated = 0, then ContainerPendingRatio = ContainerPending. The value of ContainerPendingRatio represents a number, not a percentage. This value is useful for scaling cluster resources based on container allocation behavior. Units: *Count*
AppsCompleted	The number of applications submitted to YARN that have completed. Use case: Monitor cluster progress Units: *Count*
AppsFailed	The number of applications submitted to YARN that have failed to complete. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
AppsKilled	The number of applications submitted to YARN that have been killed. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*

Metric	Description
AppsPending	The number of applications submitted to YARN that are in a pending state. Use case: Monitor cluster progress Units: *Count*
AppsRunning	The number of applications submitted to YARN that are running. Use case: Monitor cluster progress Units: *Count*
AppsSubmitted	The number of applications submitted to YARN. Use case: Monitor cluster progress Units: *Count*
Node Status	
CoreNodesRunning	The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
CoreNodesPending	The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
LiveDataNodes	The percentage of data nodes that are receiving work from Hadoop. Use case: Monitor cluster health Units: *Percent*
MRTotalNodes	The number of nodes presently available to MapReduce jobs. Equivalent to YARN metric `mapred.resourcemanager.TotalNodes`. Use ase: Monitor cluster progress Units: *Count*

Metric	Description
MRActiveNodes	The number of nodes presently running MapReduce tasks or jobs. Equivalent to YARN metric `mapred.resourcemanager.NoOfActiveNodes`. Use case: Monitor cluster progress Units: *Count*
MRLostNodes	The number of nodes allocated to MapReduce that have been marked in a LOST state. Equivalent to YARN metric `mapred.resourcemanager.NoOfLostNodes`. Use case: Monitor cluster health, Monitor cluster progress Units: *Count*
MRUnhealthyNodes	The number of nodes available to MapReduce jobs marked in an UNHEALTHY state. Equivalent to YARN metric `mapred.resourcemanager.NoOfUnhealthyNodes`. Use case: Monitor cluster progress Units: *Count*
MRDecommissionedNodes	The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state. Equivalent to YARN metric `mapred.resourcemanager.NoOfDecommissionedNodes`. Use ase: Monitor cluster health, Monitor cluster progress Units: *Count*
MRRebootedNodes	The number of nodes available to MapReduce that have been rebooted and marked in a REBOOTED state. Equivalent to YARN metric `mapred.resourcemanager.NoOfRebootedNodes`. Use case: Monitor cluster health, Monitor cluster progress Units: *Count*
IO	
S3BytesWritten	The number of bytes written to Amazon S3. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
S3BytesRead	The number of bytes read from Amazon S3. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*

Metric	Description
HDFSUtilization	The percentage of HDFS storage currently used. Use case: Analyze cluster performance Units: *Percent*
HDFSBytesRead	The number of bytes read from HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
HDFSBytesWritten	The number of bytes written to HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
MissingBlocks	The number of blocks in which HDFS has no replicas. These might be corrupt blocks. Use case: Monitor cluster health Units: *Count*
CorruptBlocks	The number of blocks that HDFS reports as corrupted. Use case: Monitor cluster health Units: *Count*
TotalLoad	The total number of concurrent data transfers. Use case: Monitor cluster health Units: *Count*
MemoryTotalMB	The total amount of memory in the cluster. Use case: Monitor cluster progress Units: *Bytes*
MemoryReservedMB	The amount of memory reserved. Use case: Monitor cluster progress Units: *Bytes*

Metric	Description
MemoryAvailableMB	The amount of memory available to be allocated. Use case: Monitor cluster progress Units: *Bytes*
YARNMemoryAvailablePercentage	The percentage of remaining memory available to YARN (YARNMemoryAvailablePercentage = MemoryAvailableMB / MemoryTotalMB). This value is useful for scaling cluster resources based on YARN memory usage.
MemoryAllocatedMB	The amount of memory allocated to the cluster. Use case: Monitor cluster progress Units: *Bytes*
PendingDeletionBlocks	The number of blocks marked for deletion. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
UnderReplicatedBlocks	The number of blocks that need to be replicated one or more times. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
DfsPendingReplicationBlocks	The status of block replication: blocks being replicated, age of replication requests, and unsuccessful replication requests. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
CapacityRemainingGB	The amount of remaining HDFS disk capacity. Use case: Monitor cluster progress, Monitor cluster health Units: *Bytes*
HBase	
HbaseBackupFailed	Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: *Count*

Metric	Description
MostRecentBackupDuration	The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters.

Use case: Monitor HBase Backups

Units: *Minutes* |
| TimeSinceLastSuccessfulBackup | The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.

Use case: Monitor HBase backups

Units: *Minutes* |

Amazon EMR Dimensions

The following dimensions are available for Amazon EMR.

Dimension	Description
JobFlowId	The same as cluster ID, which is the unique identifier of a cluster in the form `j-XXXXXXXXXXXXX`. Find this value by clicking on the cluster in the Amazon EMR console.
JobId	The identifier of a job within a cluster. You can use this to filter the metrics returned from a cluster down to those that apply to a single job within the cluster. JobId takes the form job_XXXXXXXXXXXX_XXXX.

Amazon Elasticsearch Service Metrics and Dimensions

Amazon Elasticsearch Service sends data to CloudWatch every minute. You can create alarms using Amazon Elasticsearch Service Metrics and Dimensions (p. 201). For more information, see Monitoring Cluster Metrics and Statistics with Amazon CloudWatch in the *Amazon Elasticsearch Service Developer Guide*.

Amazon Elasticsearch Service Metrics

The `AWS/ES` namespace includes the following metrics for clusters.

Metric	Description
`ClusterStatus.green`	Indicates that all index shards are allocated to nodes in the cluster.

Relevant statistics: Minimum, Maximum |

Metric	Description
ClusterStatus.yellow	Indicates that the primary shards for all indices are allocated to nodes in a cluster, but the replica shards for at least one index are not. Single node clusters always initialize with this cluster status because there is no second node to which a replica can be assigned. You can either increase your node count to obtain a green cluster status, or you can use the Elasticsearch API to set the `number_of_replicas` setting for your index to `0`. To learn more, see Configuring Amazon Elasticsearch Service Domains.

Relevant statistics: Minimum, Maximum |
| ClusterStatus.red | Indicates that the primary and replica shards of at least one index are not allocated to nodes in a cluster. To recover, you must delete the indices or restore a snapshot and then add EBS-based storage, use larger instance types, or add instances. For more information, see Red Cluster Status.

Relevant statistics: Minimum, Maximum |
| Nodes | The number of nodes in the Amazon ES cluster, including dedicated master nodes.

Relevant Statistics: Minimum, Maximum, Average |
| SearchableDocuments | The total number of searchable documents across all indices in the cluster.

Relevant statistics: Minimum, Maximum, Average |
| DeletedDocuments | The total number of documents marked for deletion across all indices in the cluster. These documents no longer appear in search results, but Elasticsearch only removes deleted documents from disk during segment merges. This metric increases after delete requests and decreases after segment merges.

Relevant statistics: Minimum, Maximum, Average |
| CPUUtilization | The maximum percentage of CPU resources used for data nodes in the cluster.

Relevant statistics: Maximum, Average |
| FreeStorageSpace | The free space, in megabytes, for nodes in the cluster. `Sum` shows total free space for the cluster. `Minimum`, `Maximum`, and `Average` show free space for individual nodes. Amazon ES throws a `ClusterBlockException` when this metric reaches `0`. To recover, you must either delete indices, add larger instances, or add EBS-based storage to existing instances. To learn more, see Recovering from a Lack of Free Storage Space

Note
`FreeStorageSpace` will always be lower than the value that the Elasticsearch `_cluster/stats` API provides. Amazon ES reserves a percentage of the storage space on each instance for internal operations.

Relevant statistics: Minimum, Maximum, Average, Sum |

Metric	Description
ClusterUsedSpace	The total used space, in megabytes, for a cluster. You can view this metric in the Amazon CloudWatch console, but not in the Amazon ES console. Relevant statistics: Minimum, Maximum
ClusterIndexWritesBlocked	Indicates whether your cluster is accepting or blocking incoming write requests. A value of 0 means that the cluster is accepting requests. A value of 1 means that it is blocking requests. Many factors can cause a cluster to begin blocking requests. Some common factors include the following: `FreeStorageSpace` is too low, `JVMMemoryPressure` is too high, or `CPUUtilization` is too high. To alleviate this issue, consider adding more disk space or scaling your cluster. Relevant statistics: Maximum **Note** You can view this metric in the Amazon CloudWatch console, but not the Amazon ES console.
JVMMemoryPressure	The maximum percentage of the Java heap used for all data nodes in the cluster. Relevant statistics: Maximum
AutomatedSnapshotFailure	The number of failed automated snapshots for the cluster. A value of 1 indicates that no automated snapshot was taken for the domain in the previous 36 hours. Relevant statistics: Minimum, Maximum
CPUCreditBalance	The remaining CPU credits available for data nodes in the cluster. A CPU credit provides the performance of a full CPU core for one minute. For more information, see CPU Credits in the *Amazon EC2 Developer Guide*. This metric is available only for the t2.micro.elasticsearch, t2.small.elasticsearch, and t2.medium.elasticsearch instance types. Relevant statistics: Minimum
KibanaHealthyNodes	A health check for Kibana. A value of 1 indicates normal behavior. A value of 0 indicates that Kibana is inaccessible. In most cases, the health of Kibana mirrors the health of the cluster. Relevant statistics: Minimum **Note** You can view this metric on the Amazon CloudWatch console, but not the Amazon ES console.
KMSKeyError	A value of 1 indicates that the KMS customer master key used to encrypt data at rest has been disabled. To restore the domain to normal operations, re-enable the key. The console displays this metric only for domains that encrypt data at rest.. Relevant statistics: Minimum, Maximum

Metric	Description
KMSKeyInaccessible	A value of 1 indicates that the KMS customer master key used to encrypt data at rest has been deleted or revoked its grants to Amazon ES. You can't recover domains that are in this state. If you have a manual snapshot, though, you can use it to migrate the domain's data to a new domain. The console displays this metric only for domains that encrypt data at rest. Relevant statistics: Minimum, Maximum
InvalidHostHeaderRequests	The number of HTTP requests made to the Elasticsearch cluster that included an invalid (or missing) host header. Valid requests include the domain endpoint as the host header value. If you see large values for this metric, check that your Elasticsearch clients include the proper host header value in their requests. Otherwise, Amazon ES might reject the requests. You can also update the domain's access policy to require signed requests. Relevant statistics: Sum
ElasticsearchRequests	The number of requests made to the Elasticsearch cluster. Relevant statistics: Sum
RequestCount	The number of requests to a domain and the HTTP response code (2xx, 3xx, 4xx, 5xx) for each request. Relevant statistics: Sum

The AWS/ES namespace includes the following metrics for dedicated master nodes.

Metric	Description
MasterCPUUtilization	The maximum percentage of CPU resources used by the dedicated master nodes. We recommend increasing the size of the instance type when this metric reaches 60 percent. Relevant statistics: Average
MasterFreeStorageSpace	This metric is not relevant and can be ignored. The service does not use master nodes as data nodes.
MasterJVMMemoryPressure	The maximum percentage of the Java heap used for all dedicated master nodes in the cluster. We recommend moving to a larger instance type when this metric reaches 85 percent. Relevant statistics: Maximum
MasterCPUCreditBalance	The remaining CPU credits available for dedicated master nodes in the cluster. A CPU credit provides the performance of a full CPU core for one minute. For more information, see CPU Credits in the *Amazon EC2 User Guide for Linux Instances*. This metric is available only for the t2.micro.elasticsearch, t2.small.elasticsearch, and t2.medium.elasticsearch instance types. Relevant statistics: Minimum

Metric	Description
MasterReachableFromNode	A health check for `MasterNotDiscovered` exceptions. A value of 1 indicates normal behavior. A value of 0 indicates that `/_cluster/ health/` is failing. Failures mean that the master node stopped or is not reachable. They are usually the result of a network connectivity issue or AWS dependency problem. Relevant statistics: Minimum **Note** You can view this metric on the Amazon CloudWatch console, but not the Amazon ES console.

The `AWS/ES` namespace includes the following metrics for EBS volumes.

Metric	Description
ReadLatency	The latency, in seconds, for read operations on EBS volumes. Relevant statistics: Minimum, Maximum, Average
WriteLatency	The latency, in seconds, for write operations on EBS volumes. Relevant statistics: Minimum, Maximum, Average
ReadThroughput	The throughput, in bytes per second, for read operations on EBS volumes. Relevant statistics: Minimum, Maximum, Average
WriteThroughput	The throughput, in bytes per second, for write operations on EBS volumes. Relevant statistics: Minimum, Maximum, Average
DiskQueueDepth	The number of pending input and output (I/O) requests for an EBS volume. Relevant statistics: Minimum, Maximum, Average
ReadIOPS	The number of input and output (I/O) operations per second for read operations on EBS volumes. Relevant statistics: Minimum, Maximum, Average
WriteIOPS	The number of input and output (I/O) operations per second for write operations on EBS volumes. Relevant statistics: Minimum, Maximum, Average

Dimensions for Amazon Elasticsearch Service Metrics

To filter the metrics, use the following dimensions.

Dimension	Description
ClientId	The AWS account ID.

Dimension	Description
`DomainName`	The name of the search domain.

Amazon Elastic Transcoder Metrics and Dimensions

When you interact with Amazon Elastic Transcoder, it sends the following metrics to CloudWatch every minute.

Elastic Transcoder Metrics

The `AWS/ElasticTranscoder` namespace includes the following metrics.

Metric	Description
`Billed HD Output`	The number of billable seconds of HD output for a pipeline. Valid Dimensions: PipelineId Unit: Seconds
`Billed SD Output`	The number of billable seconds of SD output for a pipeline. Valid Dimensions: PipelineId Unit: Seconds
`Billed Audio Output`	The number of billable seconds of audio output for a pipeline. Valid Dimensions: PipelineId Unit: Seconds
`Jobs Completed`	The number of jobs completed by this pipeline. Valid Dimensions: PipelineId Unit: Count
`Jobs Errored`	The number of jobs that failed because of invalid inputs, such as a request to transcode a file that is not in the given input bucket. Valid Dimensions: PipelineId Unit: Count
`Outputs per Job`	The number of outputs Elastic Transcoder created for a job. Valid Dimensions: PipelineId Unit: Count

Metric	Description
`Standby Time`	The number of seconds before Elastic Transcoder started transcoding a job. Valid Dimensions: PipelineId Unit: Seconds
`Errors`	The number of errors caused by invalid operation parameters, such as a request for a job status that does not include the job ID. Valid Dimensions: Operation Unit: Count
`Throttles`	The number of times that Elastic Transcoder automatically throttled an operation. Valid Dimensions: Operation Unit: Count

Dimensions for Elastic Transcoder Metrics

Elastic Transcoder metrics use the Elastic Transcoder namespace and provide metrics for the following dimension(s):

Dimension	Description
`PipelineId`	The ID of a pipeline. This dimension filters the data you request for an Elastic Transcoder pipeline.
`Operation`	This dimension filters the data you request for the APIs that Elastic Transcoder provides.

Amazon GameLift Metrics and Dimensions

Amazon GameLift Metrics for Fleets

The `AWS/GameLift` namespace includes the following metrics related to activity across a fleet or a group of fleets. The Amazon GameLift service sends metrics to CloudWatch every minute.

Instances

Metric	Description
`ActiveInstances`	Instances with ACTIVE status, which means they are running active server processes. The count includes idle instances and those that are hosting one or more game sessions. This metric measures current total instance

Metric	Description
	capacity. This metric can be used with automatic scaling. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
DesiredInstances	Target number of active instances that Amazon GameLift is working to maintain in the fleet. With automatic scaling, this value is determined based on the scaling policies currently in force. Without automatic scaling, this value is set manually. This metric is not available when viewing data for fleet metric groups. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
IdleInstances	Active instances that are currently hosting zero (0) game sessions. This metric measures capacity that is available but unused. This metric can be used with automatic scaling. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
MaxInstances	Maximum number of instances that are allowed for the fleet. A fleet's instance maximum determines the capacity ceiling during manual or automatic scaling up. This metric is not available when viewing data for fleet metric groups. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
MinInstances	Minimum number of instances allowed for the fleet. A fleet's instance minimum determines the capacity floor during manual or automatic scaling down. This metric is not available when viewing data for fleet metric groups. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum

Metric	Description
PercentIdleInstances	Percentage of all active instances that are idle (calculated as IdleInstances / ActiveInstances). This metric can be used for automatic scaling. Units: Percent Relevant CloudWatch statistics: Average, Minimum, Maximum
InstanceInterruptions	Number of spot instances that have been interrupted. Units: Count Relevant CloudWatch statistics: Sum, Average, Minimum, Maximum

Server Processes

Metric	Description
ActiveServerProcesses	Server processes with ACTIVE status, which means they are running and able to host game sessions. The count includes idle server processes and those that are hosting game sessions. This metric measures current total server process capacity. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
HealthyServerProcesses	Active server processes that are reporting healthy. This metric is useful for tracking the overall health of the fleet's game servers. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
PercentHealthyServerProcesses	Percentage of all active server processes that are reporting healthy (calculated as HealthyServerProcesses / ActiveServerProcesses). Units: Percent Relevant CloudWatch statistics: Average, Minimum, Maximum
ServerProcessAbnormalTerminations	Server processes that were shut down due to abnormal circumstances since the last report. This metric includes terminations that were initiated by the Amazon GameLift service. This occurs when a server process

Metric	Description
	stops responding, consistently reports failed health checks, or does not terminate cleanly (by calling ProcessEnding()). Units: Count Relevant CloudWatch statistics: Sum, Average, Minimum, Maximum
`ServerProcessActivations`	Server processes that successfully transitioned from ACTIVATING to ACTIVE status since the last report. Server processes cannot host game sessions until they are active. Units: Count Relevant CloudWatch statistics: Sum, Average, Minimum, Maximum
`ServerProcessTerminations`	Server processes that were shut down since the last report. This includes all server processes that transitioned to TERMINATED status for any reason, including normal and abnormal process terminations. Units: Count Relevant CloudWatch statistics: Sum, Average, Minimum, Maximum

Game Sessions

Metric	Description
`ActivatingGameSessions`	Game sessions with ACTIVATING status, which means they are in the process of starting up. Game sessions cannot host players until they are active. High numbers for a sustained period of time may indicate that game sessions are not transitioning from ACTIVATING to ACTIVE status. This metric can be used with automatic scaling. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
`ActiveGameSessions`	Game sessions with ACTIVE status, which means they are able to host players, and are hosting zero or more players. This metric measures the total number of game sessions currently being hosted. This metric can be used with automatic scaling. Units: Count

Metric	Description
	Relevant CloudWatch statistics: Average, Minimum, Maximum
`AvailableGameSessions`	Game session slots on active, healthy server processes that are not currently being used. This metric measures the number of new game sessions that could be started immediately. This metric can be used with automatic scaling. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
`PercentAvailableGameSessions`	Percentage of game session slots on all active server processes (healthy or unhealthy) that are not currently being used (calculated as `AvailableGameSessions / [ActiveGameSessions + AvailableGameSessions + unhealthy server processes]`). This metric can be used with automatic scaling. Units: Percent Relevant CloudWatch statistics: Average
`GameSessionInterruptions`	Number of game sessions on spot instances that have been interrupted. Units: Count Relevant CloudWatch statistics: Sum, Average, Minimum, Maximum

Player Sessions

Metric	Description
`CurrentPlayerSessions`	Player sessions with either ACTIVE status (player is connected to an active game session) or RESERVED status (player has been given a slot in a game session but hasn't yet connected). This metric can be used with automatic scaling. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum
`PlayerSessionActivations`	Player sessions that transitioned from RESERVED status to ACTIVE since the last report. This occurs when a player successfully connects to an active game session. Units: Count

Metric	Description
	Relevant CloudWatch statistics: Sum, Average, Minimum, Maximum

Amazon GameLift Metrics for Queues

The `GameLift` namespace includes the following metrics related to activity across a game session placement queue. The Amazon GameLift service sends metrics to CloudWatch every minute.

Metric	Description
`AverageWaitTime`	Average amount of time that game session placement requests in the queue with status PENDING have been waiting to be fulfilled. Units: Seconds Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
`FirstChoiceNotViable`	Game sessions that were successfully placed but NOT in the first-choice fleet, because that fleet was considered not viable (such as a spot fleet with a high interruption rate). The first-choice fleet is either the first fleet listed in the queue or—when a placement request includes player latency data—it is the first fleet chosen by prioritization. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
`FirstChoiceOutOfCapacity`	Game sessions that were successfully placed but NOT in the first-choice fleet, because that fleet had no available resources. The first-choice fleet is either the first fleet listed in the queue or—when a placement request includes player latency data —it is the first fleet chosen by prioritization. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
`LowestLatencyPlacement`	Game sessions that were successfully placed in a region that offers the queue's lowest possible latency for the players. This metric is emitted only when player latency data is included in the placement request, which triggers prioritization. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum

Metric	Description
LowestPricePlacement	Game sessions that were successfully placed in a fleet with the queue's lowest possible price for the chosen region. (prioritization first chooses the region with the lowest latency for the players and then finds the lowest cost fleet within that region.) This fleet can be either a spot fleet or an on-demand instance if the queue has no spot instances. This metric is emitted only when player latency data is included in the placement request. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
Placement <region name>	Game sessions that are successfully placed in fleets located in the specified region. This metric breaks down the `PlacementsSucceeded` metric by region. Units: Count Relevant CloudWatch statistics: Sum
PlacementsCanceled	Game session placement requests that were canceled before timing out since the last report. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
PlacementsFailed	Game session placement requests that failed for any reason since the last report. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
PlacementsStarted	New game session placement requests that were added to the queue since the last report. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
PlacementsSucceeded	Game session placement requests that resulted in a new game session since the last report. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum

Metric	Description
PlacementsTimedOut	Game session placement requests that reached the queue's timeout limit without being fulfilled since the last report. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
QueueDepth	Number of game session placement requests in the queue with status PENDING. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum

Amazon GameLift Metrics for Matchmaking

The `GameLift` namespace includes metrics on matchmaking activity for matchmaking configurations and matchmaking rules. The Amazon GameLift service sends metrics to CloudWatch every minute.

For more information on the sequence of matchmaking activity, see How Amazon GameLift FlexMatch Works.

Matchmaking Configurations

Metric	Description
CurrentTickets	Matchmaking requests currently being processed or waiting to be processed. Units: Count Relevant CloudWatch statistics: Average, Minimum, Maximum, Sum
MatchAcceptancesTimedOut	For matchmaking configurations that require acceptance, the potential matches that timed out during acceptance since the last report. Units: Count Relevant CloudWatch statistics: Sum
MatchesAccepted	For matchmaking configurations that require acceptance, the potential matches that were accepted since the last report. Units: Count Relevant CloudWatch statistics: Sum
MatchesCreated	Potential matches that were created since the last report.

Metric	Description
	Units: Count
	Relevant CloudWatch statistics: Sum
MatchesPlaced	Matches that were successfully placed into a game session since the last report.
	Units: Count
	Relevant CloudWatch statistics: Sum
MatchesRejected	For matchmaking configurations that require acceptance, the potential matches that were rejected by at least one player since the last report.
	Units: Count
	Relevant CloudWatch statistics: Sum
PlayersStarted	Players in matchmaking tickets that were added since the last report.
	Units: Count
	Relevant CloudWatch statistics: Sum
TicketsFailed	Matchmaking requests that resulted in failure since the last report.
	Units: Count
	Relevant CloudWatch statistics: Sum
TicketsStarted	New matchmaking requests that were created since the last report.
	Units: Count
	Relevant CloudWatch statistics: Sum
TicketsTimedOut	Matchmaking requests that reached the timeout limit since the last report.
	Units: Count
	Relevant CloudWatch statistics: Sum
TimeToMatch	For matchmaking requests that were put into a potential match before the last report, the amount of time between ticket creation and potential match creation.
	Units: Seconds
	Relevant CloudWatch statistics: Data Samples, Average, Minimum, Maximum, p99

Metric	Description
`TimeToTicketCancel`	For matchmaking requests that were canceled before the last report, the amount of time between ticket creation and cancellation. Units: Seconds Relevant CloudWatch statistics: Data Samples, Average, Minimum, Maximum, p99
`TimeToTicketSuccess`	For matchmaking requests that succeeded before the last report, the amount of time between ticket creation and successful match placement. Units: Seconds Relevant CloudWatch statistics: Data Samples, Average, Minimum, Maximum, p99

Matchmaking Rules

Metric	Description
`RuleEvaluationsPassed`	Rule evaluations during the matchmaking process that passed since the last report. This metric is limited to the top 50 rules. Units: Count Relevant CloudWatch statistics: Sum
`RuleEvaluationsFailed`	Rule evaluations during matchmaking that failed since the last report. This metric is limited to the top 50 rules. Units: Count Relevant CloudWatch statistics: Sum

Dimensions for Amazon GameLift Metrics

Amazon GameLift supports filtering metrics by the following dimensions.

Dimension	Description
`FleetId`	Unique identifier for a single fleet. This dimension is used with all metrics for instances, server processes, game sessions, and player sessions. It is not used with metrics for queues and matchmaking.
`MetricGroup`	Unique identifier for a collection of fleets. A fleet is included in a fleet metric group by adding the metric group name to the fleet's attributes (see UpdateFleetAttributes()). This dimension is used with all metrics for instances, server processes, game sessions,

Dimension	Description
	and player sessions. It is not used with metrics for queues and matchmaking.
QueueName	Unique identifier for a single queue. This dimension is used with metrics for game session placement queues only.
MatchmakingConfigurationName	Unique identifier for a single matchmaking configuration. This dimension is used with metrics for matchmaking only.
MatchmakingConfigurationName-RuleName	Unique identifier for the intersect of a matchmaking configuration and a matchmaking rule. This dimension is used with metrics for matchmaking only.
InstanceType	Unique identifier for an EC2 instance type designation, such as "C4.large". This dimension is used with metrics for spot instances only.
OperatingSystem	Unique identifier for the operating system of an instance. This dimension is used with metrics for spot instances only.

AWS Glue Metrics and Dimensions

AWS Glue sends metrics to CloudWatch. For more information, see Monitoring AWS Glue Using CloudWatch Metrics in the AWS Glue Developer Guide.

AWS Glue Metrics

AWS Glue profiles and sends the following metrics to CloudWatch every 30 seconds, and the AWS Glue Metrics Dashboard report them once a minute:

Metric	Description
glue.driver.aggregate.bytesRead	The number of bytes read from all data sources by all completed Spark tasks running in all executors..
	Valid dimensions: JobName (the name of the AWS Glue Job), JobRunId (the JobRun ID. or ALL), and Type (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Bytes
	Can be used to monitor:
	• Bytes read. • Job progress. • JDBC data sources. • Job Bookmark Issues.

Metric	Description
	• Variance across Job Runs.
	This metric can be used the same way as the `glue.ALL.s3.filesystem.read_bytes` metric, with the difference that this metric is updated at the end of a Spark task and captures non-S3 data sources as well.
`glue.driver.aggregate.elapsedTime`	The ETL elapsed time in milliseconds (does not include the job bootstrap times).
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Milliseconds
	Can be used to determine how long it takes a job run to run on average.
	Some ways to use the data:
	• Set alarms for stragglers.
	• Measure variance across job runs.
`glue.driver.aggregate.numCompletedStages`	The number of completed stages in the job.
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Count
	Can be used to monitor:
	• Job progress.
	• Per-stage timeline of job execution,when correlated with other metrics.
	Some ways to use the data:
	• Identify demanding stages in the execution of a job.
	• Set alarms for correlated spikes (demanding stages) across job runs.

Metric	Description
`glue.driver.aggregate.numCompletedTasks`	The number of completed tasks in the job.
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Count
	Can be used to monitor:
	- Job progress. - Parallelism within a stage.
`glue.driver.aggregate.numFailedTasks`	The number of failed tasks.
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Count
	Can be used to monitor:
	- Data abnormalities that cause job tasks to fail. - Cluster abnormalities that cause job tasks to fail. - Script abnormalities that cause job tasks to fail.
	The data can be used to set alarms for increased failures that might suggest abnormalities in data, cluster or scripts.

Metric	Description
`glue.driver.aggregate.numKilledTasks`	The number of tasks killed.
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Count
	Can be used to monitor:
	• Abnormalities in Data Skew that result in exceptions (OOMs) that kill tasks. • Script abnormalities that result in exceptions (OOMs) that kill tasks.
	Some ways to use the data:
	• Set alarms for increased failures indicating data abnormalities. • Set alarms for increased failures indicating cluster abnormalities. • Set alarms for increased failures indicating script abnormalities.
`glue.driver.aggregate.recordsRead`	The number of records read from all data sources by all completed Spark tasks running in all executors.
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count).
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation.
	Unit: Count
	Can be used to monitor:
	• Records read. • Job progress. • JDBC data sources. • Job Bookmark Issues. • Skew in Job Runs over days.
	This metric can be used in a similar way to the `glue.ALL.s3.filesystem.read_bytes` metric, with the difference that this metric is updated at the end of a Spark task.

Metric	Description
`glue.driver.aggregate.shuffleBytesWritten`	The number of bytes written by all executors to shuffle data between them since the previous report (aggregated by the AWS Glue Metrics Dashboard as the number of bytes written for this purpose during the previous minute). Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count). Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation. Unit: Bytes Can be used to monitor: Data shuffle in jobs (large joins, groupBy, repartition, coalesce). Some ways to use the data: • Repartition or decompress large input files before further processing. • Repartition data more uniformly to avoid hot keys. • Pre-filter data before joins or groupBy operations.
`glue.driver.aggregate.shuffleLocalBytesRead`	The number of bytes read by all executors to shuffle data between them since the previous report (aggregated by the AWS Glue Metrics Dashboard as the number of bytes read for this purpose during the previous minute). Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (count). Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard, a SUM statistic is used for aggregation. Unit: Bytes Can be used to monitor: Data shuffle in jobs (large joins, groupBy, repartition, coalesce). Some ways to use the data: • Repartition or decompress large input files before further processing. • Repartition data more uniformly using hot keys. • Pre-filter data before joins or groupBy operations.

Metric	Description
`glue.driver.BlockManager.disk.diskSpaceUsed_MB`	The number of megabytes of disk space used across all executors. Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (gauge). Valid Statistics: Average. This is a Spark metric, reported as an absolute value. Unit: Megabytes Can be used to monitor: - Disk space used for blocks that represent cached RDD partitions. - Disk space used for blocks that represent intermediate shuffle outputs. - Disk space used for blocks that represent broadcasts. Some ways to use the data: - Identify job failures due to increased disk usage. - Identify large partitions resulting in spilling or shuffling. - Increase provisioned DPU capacity to correct these issues.

`glue.driver.BlockManager.disk.diskSpaceUsed_MB` The number of

Metric	Description
`glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`	The number of actively running job executors. Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (gauge).

Valid Statistics: Average. This is a Spark metric, reported as an absolute value.

Unit: Count

Can be used to monitor:

- Job activity.
- Straggling executors (with a few executors running only)
- Current executor-level parallelism.

Some ways to use the data:

- Repartition or decompress large input files beforehand if cluster is under-utilized.
- Identify stage or job execution delays due to straggler scenarios.
- • Compare with numberMaxNeededExecutors to understand backlog for provisioning more DPUs.

Metric	Description
glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors	The number of maximum (actively running and pending) job executors needed to satisfy the current load.
	Valid dimensions: JobName (the name of the AWS Glue Job), JobRunId (the JobRun ID. or ALL), and Type (gauge).
	Valid Statistics: Maximum. This is a Spark metric, reported as an absolute value.
	Unit: Count
	Can be used to monitor:
	• Job activity.
	• Current executor-level parallelism and backlog of pending tasks not yet scheduled because of unavailable executors due to DPU capacity or killed/ failed executors.
	Some ways to use the data:
	• Identify pending/backlog of scheduling queue.
	• Identify stage or job execution delays due to straggler scenarios.
	• Compare with numberAllExecutors to understand backlog for provisioning more DPUs.
	• Increase provisioned DPU capacity to correct the pending executor backlog.

Metric	Description

`glue.driver.jvm.heap.usage`
 	
`glue.`*executorId*`.jvm.heap.usage`
 	
`glue.ALL.jvm.heap.usage`
	The fraction of memory used by the JVM heap for this driver (scale: 0-1) for driver, executor identified by executorId, or ALL executors.
	
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (gauge).
	
	Valid Statistics: Average. This is a Spark metric, reported as an absolute value.
	
	Unit: Percentage
	
	Can be used to monitor:
	
	• Driver out-of-memory conditions (OOM) using `glue.driver.jvm.heap.usage`.
	• Executor out-of-memory conditions (OOM) using `glue.ALL.jvm.heap.usage`.
	
	Some ways to use the data:
	
	• Identify memory-consuming executor ids and stages.
	• Identify straggling executor ids and stages.
	• Identify a driver out-of-memory condition (OOM).
	• Identify an executor out-of-memory condition (OOM) and obtain the corresponding executor ID so as to be able to get a stack trace from the executor log.
	• Identify files or partitions that may have data skew resulting in stragglers or out-of-memory conditions (OOMs).

Metric	Description

`glue.driver.jvm.heap.used`
 	
`glue.`*executorId*`.jvm.heap.used`
 	
`glue.ALL.jvm.heap.used`
	The number of memory bytes used by the JVM heap for the driver, the executor identified by *executorId*, or ALL executors.
	
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (gauge).
	
	Valid Statistics: Average. This is a Spark metric, reported as an absolute value.
	
	Unit: Bytes
	
	Can be used to monitor:
	
	• Driver out-of-memory conditions (OOM).
	• Executor out-of-memory conditions (OOM).
	
	Some ways to use the data:
	
	• Identify memory-consuming executor ids and stages.
	• Identify straggling executor ids and stages.
	• Identify a driver out-of-memory condition (OOM).
	• Identify an executor out-of-memory condition (OOM) and obtain the corresponding executor ID so as to be able to get a stack trace from the executor log.
	• Identify files or partitions that may have data skew resulting in stragglers or out-of-memory conditions (OOMs).

Metric	Description

`glue.driver.s3.filesystem.read_bytes`
 	
`glue.executorId.s3.filesystem.read_bytes`
 	
`glue.ALL.s3.filesystem.read_bytes`
	The number of bytes read from Amazon S3 by the driver, an executor identified by *executorId*, or ALL executors since the previous report (aggregated by the AWS Glue Metrics Dashboard as the number of bytes read during the previous minute).
	
	Valid dimensions: `JobName`, `JobRunId`, and `Type` (gauge).
	
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard a SUM statistic is used for aggregation. The area under the curve on the AWS Glue Metrics Dashboard can be used to visually compare bytes read by two different job runs.
	
	Unit: Bytes.
	
	Can be used to monitor:
	
	• ETL data movement.
	• Job progress.
	• Job bookmark issues (data processed, reprocessed, and skipped).
	• Comparison of reads to ingestion rate from external data sources.
	• Variance across job runs.
	
	Resulting data can be used for:
	
	• DPU capacity planning.
	• Setting alarms for large spikes or dips in data read for job runs and job stages.

Metric	Description

`glue.driver.s3.filesystem.write_bytes`
 	
`glue.`*executorId*`.s3.filesystem.write_bytes`
 	
`glue.ALL.s3.filesystem.write_bytes`
	The number of bytes written to Amazon S3 by the driver, an executor identified by *executorId*, or ALL executors since the previous report (aggregated by the AWS Glue Metrics Dashboard as the number of bytes written during the previous minute).
	
	Valid dimensions: `JobName`, `JobRunId`, and `Type` (gauge).
	
	Valid Statistics: SUM. This metric is a delta value from the last reported value, so on the AWS Glue Metrics Dashboard a SUM statistic is used for aggregation. The area under the curve on the AWS Glue Metrics Dashboard can be used to visually compare bytes written by two different job runs.
	
	Unit: Bytes
	
	Can be used to monitor:
	
	• ETL data movement.
	• Job progress.
	• Job bookmark issues (data processed, reprocessed, and skipped).
	• Comparison of reads to ingestion rate from external data sources.
	• Variance across job runs.
	
	Some ways to use the data:
	
	• DPU capacity planning.
	• Setting alarms for large spikes or dips in data read for job runs and job stages.

Metric	Description

`glue.driver.system.cpuSystemLoad`
 	
`glue.`*`executorId`*`.system.cpuSystemLoad`
 	
`glue.ALL.system.cpuSystemLoad`
	The fraction of CPU system load used (scale: 0-1) by the driver, an executor identified by *executorId*, or ALL executors.
	
	Valid dimensions: `JobName` (the name of the AWS Glue Job), `JobRunId` (the JobRun ID. or `ALL`), and `Type` (gauge).
	
	Valid Statistics: Average. This metric is reported as an absolute value.
	
	Unit: Percentage
	
	Can be used to monitor:
	
	• Driver CPU load.
	• Executor CPU load.
	• Detecting CPU-bound or IO-bound executors or stages in a Job.
	
	Some ways to use the data:
	
	• DPU capacity Planning along with IO Metrics (Bytes Read/Shuffle Bytes, Task Parallelism) and the number of maximum needed executors metric.
	• Identify the CPU/IO-bound ratio. This allows for repartitionioning and increasing provisioned capacity for long-running jobs with splittable datasets having lower CPU utilization.

Dimensions for AWS Glue Metrics

AWS Glue metrics use the AWS Glue namespace and provide metrics for the following dimensions:

Dimension	Description
`JobName`	This dimension filters for metrics of all job runs of a specific AWS Glue job.
`JobRunId`	This dimension filters for metrics of a specific AWS Glue job run by a JobRun ID, or `ALL`.
`Type`	This dimension filters for metrics by either `count` (an aggregate number) or `gauge` (a value at a point in time).

Amazon Inspector Metrics

For information about the Amazon Inspector metrics that you can use with CloudWatch, see Monitoring Amazon Inspector Using CloudWatch in the Amazon Inspector User Guide.

AWS IoT Metrics and Dimensions

When you interact with AWS IoT, it sends the following metrics to CloudWatch every minute, in the `AWS/IoT` namespace.

AWS IoT Metrics

The `AWS/IoT` namespace includes the following metrics. AWS IoT sends the following metrics to CloudWatch once per received request.

IoT Metrics

Metric	Description
RulesExecuted	The number of AWS IoT rules executed.
NumLogEventsFailedToPublishThrottled	The number of log events within the batch that have failed to publish due to throttling errors.
NumLogBatchesFailedToPublishThrottled	The singular batch of log events that has failed to publish due to throttling errors.

Rule Metrics

Metric	Description
TopicMatch	The number of incoming messages published on a topic on which a rule is listening. The `RuleName` dimension contains the name of the rule.
ParseError	The number of JSON parse errors that occurred in messages published on a topic on which a rule is listening. The `RuleName` dimension contains the name of the rule.

Rule Action Metrics

Metric	Description
Success	The number of successful rule action invocations. The `RuleName` dimension contains the name of the rule that specifies the action. The `ActionType` dimension contains the type of action that was invoked.
Failure	The number of failed rule action invocations. The `RuleName` dimension contains the name of the rule that specifies the action. The `RuleName` dimension contains the name of the rule that specifies the action. The `ActionType` dimension contains the type of action that was invoked.

Message Broker Metrics

Metric	Description
Connect.AuthError	The number of connection requests that could not be authorized by the message broker. The `Protocol` dimension contains the protocol used to send the `CONNECT.` message.
Connect.ClientError	The number of connection requests rejected because the MQTT message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `CONNECT.` message.
Connect.ServerError	The number of connection requests that failed because an internal error occurred. The Protocol dimension contains the protocol used to send the `CONNECT` message.
Connect.Success	The number of successful connections to the message broker. The `Protocol` dimension contains the protocol used to send the `CONNECT` message.
Connect.Throttle	The number of connection requests that were throttled because the client exceeded the allowed connect request rate. The `Protocol` dimension contains the protocol used to send the `CONNECT` message.
Ping.Success	The number of ping messages received by the message broker. The `Protocol` dimension contains the protocol used to send the ping message.
PublishIn.AuthError	The number of publish requests the message broker was unable to authorize. The `Protocol` dimension contains the protocol used to publish the message.
PublishIn.ClientError	The number of publish requests rejected by the message broker because the message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to publish the message.
PublishIn.ServerError	The number of publish requests the message broker failed to process because an internal error occurred. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishIn.Success	The number of publish requests successfully processed by the message broker. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishIn.Throttle	The number of publish request that were throttled because the client exceeded the allowed inbound message rate. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishOut.AuthError	The number of publish requests made by the message broker that could not be authorized by AWS IoT. The

Metric	Description
	`Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishOut.ClientError	The number of publish requests made by the message broker that were rejected because the message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
PublishOut.Success	The number of publish requests successfully made by the message broker. The `Protocol` dimension contains the protocol used to send the `PUBLISH` message.
Subscribe.AuthError	The number of subscription requests made by a client that could not be authorized. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.ClientError	The number of subscribe requests that were rejected because the `SUBSCRIBE` message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.ServerError	The number of subscribe requests that were rejected because an internal error occurred. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.Success	The number of subscribe requests that were successfully processed by the message broker. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Subscribe.Throttle	The number of subscribe requests that were throttled because the client exceeded the allowed subscribe request rate. The `Protocol` dimension contains the protocol used to send the `SUBSCRIBE` message.
Unsubscribe.ClientError	The number of unsubscribe requests that were rejected because the `UNSUBSCRIBE` message did not meet the requirements defined in AWS IoT Limits. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.
Unsubscribe.ServerError	The number of unsubscribe requests that were rejected because an internal error occurred. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.
Unsubscribe.Success	The number of unsubscribe requests that were successfully processed by the message broker. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.

Metric	Description
Unsubscribe.Throttle	The number of unsubscribe requests that were rejected because the client exceeded the allowed unsubscribe request rate. The `Protocol` dimension contains the protocol used to send the `UNSUBSCRIBE` message.

Note
The message broker metrics are displayed in the AWS IoT console under **Protocol Metrics**.

Device Shadow Metrics

Metric	Description
DeleteThingShadow.Accepted	The number of DeleteThingShadow requests processed successfully. The `Protocol` dimension contains the protocol used to make the request.
GetThingShadow.Accepted	The number of GetThingShadow requests processed successfully. The `Protocol` dimension contains the protocol used to make the request.
UpdateThingShadow.Accepted	The number of UpdateThingShadow requests processed successfully. The `Protocol` dimension contains the protocol used to make the request.

Note
The device shadow metrics are displayed in the AWS IoT console under **Protocol Metrics**.

Jobs Metrics

Metric	Description
ServerError	The number of server errors generated while executing the job. The `JobId` dimension contains the ID of the job.
ClientError	The number of client errors generated while executing the job. The `JobId` dimension contains the ID of the job.
QueuedJobExecutionTotalCount	The total number of job executions whose status is `QUEUED` for the given job. The `JobId` dimension contains the ID of the job.
InProgressJobExecutionTotalCount	The total number of job executions whose status is `IN_PROGRESS` for the given job. The `JobId` dimension contains the ID of the job.
FailedJobExecutionTotalCount	The total number of job executions whose status is `FAILED` for the given job. The `JobId` dimension contains the ID of the job.
SuccededJobExecutionTotalCount	The total number of job executions whose status is `SUCCESS` for the given job. The `JobId` dimension contains the ID of the job.

Metric	Description
CanceledJobExecutionTotalCount	The total number of job executions whose status is `CANCELED` for the given job. The `JobId` dimension contains the ID of the job.
RejectedJobExecutionTotalCount	The total number of job executions whose status is `REJECTED` for the given job. The `JobId` dimension contains the ID of the job.
RemovedJobExecutionTotalCount	The total number of job executions whose status is `REMOVED` for the given job. The `JobId` dimension contains the ID of the job.
QueuedJobExecutionCount	The number of job executions whose status has changed to `QUEUED` within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The `JobId` dimension contains the ID of the job.
InProgressJobExecutionCount	The number of job executions whose status has changed to `IN_PROGRESS` within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The `JobId` dimension contains the ID of the job.
FailedJobExecutionCount	The number of job executions whose status has changed to `FAILED` within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The `JobId` dimension contains the ID of the job.
SuccededJobExecutionCount	The number of job executions whose status has changed to `SUCCESS` within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The `JobId` dimension contains the ID of the job.
CanceledJobExecutionCount	The number of job executions whose status has changed to `CANCELED` within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The `JobId` dimension contains the ID of the job.
RejectedJobExecutionCount	The number of job executions whose status has changed to `REJECTED` within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The `JobId` dimension contains the ID of the job.

Metric	Description
RemovedJobExecutionCount	The number of job executions whose status has changed to REMOVED within a time period that is determined by CloudWatch. (For more information about CloudWatch metrics, see Amazon CloudWatch Metrics.) The JobId dimension contains the ID of the job.

Device Defender Audit Metrics

Metric	Description
NonCompliantResources	The number of resources that were found to be non-compliant with a check. The system will report the number of resources which were out of compliance for each check of each audit performed.
ResourcesEvaluated	The number of resources that were evaluated for compliance. The system will report the number of resources which were evaluated for each check of each audit performed.

Device Defender Detect Metrics

Metric	Description
Violations	The number of new violations of security profile behaviors that have been found since the last time an evaluation was performed. The system will report the number of new violations for the account, for a specific security profile, and for a specific behavior of a specific security profile.
ViolationsCleared	The number of violations of security profile behaviors that have resolved since the last time an evaluation was performed. The system will report the number of resolved violations for the account, for a specific security profile, and for a specific behavior of a specific security profile.
ViolationsInvalidated	The number of violations of security profile behaviors for which information is no longer available since the last time an evaluation was performed (because the reporting device stopped reporting, or is no longer being monitored for some reason). The system will report the number of invalidated violations for the entire account, for a specific security profile, and for a specific behavior of a specific security profile.

Dimensions for Metrics

Metrics use the namespace and provide metrics for the following dimension(s):

Dimension	Description
ActionType	The action type specified by the rule that triggered by the request.
Protocol	The protocol used to make the request. Valid values are: MQTT or HTTP
RuleName	The name of the rule triggered by the request.
JobId	The ID of the job whose progress or message connection success/failure is being monitored.
TaskType	The type of Device Defender Audit whose check results are being monitored. One of "SCHEDULED_AUDIT_TASK" or "ON_DEMAND_AUDIT_TASK".
CheckName	The name of the Device Defender audit check whose results are being monitored.
ScheduledAuditName	The name of the Device Defender scheduled audit whose check results are being monitored. This will have the value "OnDemand" if the results reported are for an audit that was performed on-demand.
SecurityProfileName	The name of the Device Defender Detect security profile whose behaviors are being monitored.
BehaviorName	The name of the Device Defender Detect security profile behavior which is being monitored.

AWS IoT Analytics Metrics and Dimensions

AWS IoT Analytics sends the following metrics to CloudWatch.

AWS IoT Analytics Metrics

AWS IoT Analytics sends the following metrics to CloudWatch once per received request.

AWS IoT Analytics Metrics

Metric	Description
ActionExecution	The number of actions executed.
ActivityExecutionError	The number of errors generated while executing the pipeline activity.
IncomingMessages	The number of messages coming into the channel.

Dimensions for Metrics

Metrics use the namespace and provide metrics for the following dimension(s):

Dimension	Description
ActionType	The type of action that is being monitored.
ChannelName	The name of the channel that is being monitored.
DatasetName	The name of the data set that is being monitored.
DatastoreName	The name of the data store that is being monitored.
PipelineActivityName	The name of the pipeline activity that is being monitored.
PipelineActivityType	The type of the pipeline activity that is being monitored.
PipelineName	The name of the pipeline that is being monitored.

Amazon Kinesis Data Analytics Metrics

Kinesis Data Analytics sends metrics to CloudWatch. For more information, see Monitoring with Amazon CloudWatch Metrics in the *Amazon Kinesis Data Analytics Developer Guide*.

Metrics

The `AWS/KinesisAnalytics` namespace includes the following metrics.

Metric	Description
`Bytes`	The number of bytes read (per input stream) or written (per output stream). Levels: Per input stream and per output stream
`KPUs`	The number of Kinesis Processing Units that are used to run your stream processing application. The average number of KPUs used each hour determines the billing for your application. Levels: Application-level
`MillisBehindLatest`	Indicates how far behind from the current time an application is reading from the streaming source. Levels: Application-level
`Records`	The number of records read (per input stream) or written (per output stream). Levels: Per input stream and per output stream
`Success`	1 for each successful delivery attempt to the destination configured for your application; 0 for every failed delivery attempt. The average value of this metric indicates how many successful deliveries are performed. Levels: Per destination.

Metric	Description
InputProcessing.Duration	The time taken for each Lambda function invocation performed by Kinesis Data Analytics.
	Levels: Per input stream
InputProcessing.OkRecords	The number of records returned by a Lambda function that were marked with Ok status.
	Levels: Per input stream
InputProcessing.OkBytes	The sum of bytes of the records returned by a Lambda function that were marked with Ok status.
	Levels: Per input stream
InputProcessing.DroppedRecords	The number of records returned by a Lambda function that were marked with Dropped status.
	Levels: Per input stream
InputProcessing.ProcessingFailedRecords	The number of records returned by a Lambda function that were marked with ProcessingFailed status.
	Levels: Per input stream
InputProcessing.Success	The number of successful Lambda invocations by Kinesis Data Analytics.
	Levels: Per input stream
LambdaDelivery.OkRecords	The number of records returned by a Lambda function that were marked with Ok status.
	Levels: Per Lambda destination
LambdaDelivery.DeliveryFailedRecords	The number of records returned by a Lambda function that were marked with DeliveryFailed status.
	Levels: Per Lambda destination
LambdaDelivery.Duration	The time taken for each Lambda function invocation performed by Kinesis Data Analytics.
	Levels: Per Lambda destination

Dimensions for Metrics

Amazon Kinesis Data Analytics provides metrics for the following dimensions.

Dimension	Description
Flow	Per input stream: Input
	Per output stream: Output
Id	Per input stream: Input Id

Dimension	Description
	Per output stream: Output Id

Amazon Kinesis Data Firehose Metrics

Kinesis Data Firehose sends metrics to CloudWatch. For more information, see Monitoring with Amazon CloudWatch Metrics in the *Amazon Kinesis Data Firehose Developer Guide*.

Service-level CloudWatch Metrics

The `AWS/Firehose` namespace includes the following service-level metrics.

Metric	Description
`BackupToS3.Bytes`	The number of bytes delivered to Amazon S3 for backup over the specified time period. Kinesis Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations. Units: Bytes
`BackupToS3.DataFreshness`	Age (from getting into Kinesis Data Firehose to now) of the oldest record in Kinesis Data Firehose. Any record older than this age has been delivered to the Amazon S3 bucket for backup. Kinesis Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations. Units: Seconds
`BackupToS3.Records`	The number of records delivered to Amazon S3 for backup over the specified time period. Kinesis Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations. Units: Count
`BackupToS3.Success`	Sum of successful Amazon S3 put commands for backup over sum of all Amazon S3 backup put commands. Kinesis Data Firehose emits this metric when data transformation is enabled for Amazon S3 or Amazon Redshift destinations.
`DataReadFromKinesisStream.Bytes`	When the data source is a Kinesis data stream, this metric indicates the number of bytes read from that data stream. This number includes rereads due to failovers. Units: Bytes
`DataReadFromKinesisStream.Records`	When the data source is a Kinesis data stream, this metric indicates the number of records read from that data stream. This number includes rereads due to failovers. Units: Count
`DeliveryToElasticsearch.Bytes`	The number of bytes indexed to Amazon ES over the specified time period.

Metric	Description
	Units: Bytes
`DeliveryToElasticsearch.Records`	The number of records indexed to Amazon ES over the specified time period. Units: Count
`DeliveryToElasticsearch.Success`	The sum of the successfully indexed records over the sum of records that were attempted.
`DeliveryToRedshift.Bytes`	The number of bytes copied to Amazon Redshift over the specified time period. Units: Bytes
`DeliveryToRedshift.Records`	The number of records copied to Amazon Redshift over the specified time period. Units: Count
`DeliveryToRedshift.Success`	The sum of successful Amazon Redshift COPY commands over the sum of all Amazon Redshift COPY commands.
`DeliveryToS3.Bytes`	The number of bytes delivered to Amazon S3 over the specified time period. Units: Bytes
`DeliveryToS3.DataFreshness`	The age (from getting into Kinesis Data Firehose to now) of the oldest record in Kinesis Data Firehose. Any record older than this age has been delivered to the S3 bucket. Units: Seconds
`DeliveryToS3.Records`	The number of records delivered to Amazon S3 over the specified time period. Units: Count
`DeliveryToS3.Success`	The sum of successful Amazon S3 put commands over the sum of all Amazon S3 put commands.
`DeliveryToSplunk.Bytes`	The number of bytes delivered to Splunk over the specified time period. Units: Bytes
`DeliveryToSplunk.DataAckLatency`	The approximate duration it takes to receive an acknowledgement from Splunk after Kinesis Data Firehose sends it data. The increasing or decreasing trend for this metric is more useful than the absolute approximate value. Increasing trends can indicate slower indexing and acknowledgement rates from Splunk indexers. Units: Seconds

Metric	Description
`DeliveryToSplunk.DataFreshness`	Age (from getting into Kinesis Data Firehose to now) of the oldest record in Kinesis Data Firehose. Any record older than this age has been delivered to Splunk. Units: Seconds
`DeliveryToSplunk.Records`	The number of records delivered to Splunk over the specified time period. Units: Count
`DeliveryToSplunk.Success`	The sum of the successfully indexed records over the sum of records that were attempted.
`IncomingBytes`	The number of bytes ingested into the Kinesis Data Firehose stream over the specified time period. Units: Bytes
`IncomingRecords`	The number of records ingested into the Kinesis Data Firehose stream over the specified time period. Units: Count
`KinesisMillisBehindLatest`	When the data source is a Kinesis data stream, this metric indicates the number of milliseconds that the last read record is behind the newest record in the Kinesis data stream. Units: Millisecond

API-Level CloudWatch Metrics

The `AWS/Firehose` namespace includes the following API-level metrics.

Metric	Description
`DescribeDeliveryStream.Latency`	The time taken per `DescribeDeliveryStream` operation, measured over the specified time period. Units: Milliseconds
`DescribeDeliveryStream.Requests`	The total number of `DescribeDeliveryStream` requests. Units: Count
`ListDeliveryStreams.Latency`	The time taken per `ListDeliveryStream` operation, measured over the specified time period. Units: Milliseconds
`ListDeliveryStreams.Requests`	The total number of `ListFirehose` requests. Units: Count

Metric	Description
`PutRecord.Bytes`	The number of bytes put to the Kinesis Data Firehose delivery stream using `PutRecord` over the specified time period. Units: Bytes
`PutRecord.Latency`	The time taken per `PutRecord` operation, measured over the specified time period. Units: Milliseconds
`PutRecord.Requests`	The total number of `PutRecord` requests, which is equal to total number of records from `PutRecord` operations. Units: Count
`PutRecordBatch.Bytes`	The number of bytes put to the Kinesis Data Firehose delivery stream using `PutRecordBatch` over the specified time period. Units: Bytes
`PutRecordBatch.Latency`	The time taken per `PutRecordBatch` operation, measured over the specified time period. Units: Milliseconds
`PutRecordBatch.Records`	The total number of records from `PutRecordBatch` operations. Units: Count
`PutRecordBatch.Requests`	The total number of `PutRecordBatch` requests. Units: Count
`ThrottledDescribeStream`	The total number of times the `DescribeStream` operation is throttled when the data source is a Kinesis data stream. Units: Count
`ThrottledGetRecords`	The total number of times the `GetRecords` operation is throttled when the data source is a Kinesis data stream. Units: Count
`ThrottledGetShardIterator`	The total number of times the `GetShardIterator` operation is throttled when the data source is a Kinesis data stream. Units: Count
`UpdateDeliveryStream.Latency`	The time taken per `UpdateDeliveryStream` operation, measured over the specified time period. Units: Milliseconds

Metric	Description
`UpdateDeliveryStream.Requests`	The total number of `UpdateDeliveryStream` requests. Units: Count

Data Transformation CloudWatch Metrics

If data transformation with Lambda is enabled, the `AWS/Firehose` namespace includes the following metrics.

Metric	Description
`ExecuteProcessing.Duration`	The time it takes for each Lambda function invocation performed by Kinesis Data Firehose. Units: Milliseconds
`ExecuteProcessing.Success`	The sum of the successful Lambda function invocations over the sum of the total Lambda function invocations.
`SucceedProcessing.Records`	The number of successfully processed records over the specified time period. Units: Count
`SucceedProcessing.Bytes`	The number of successfully processed bytes over the specified time period. Units: Bytes

Amazon Kinesis Data Streams Metrics and Dimensions

Kinesis Data Streams sends metrics to CloudWatch at two levels; the stream level and, optionally, the shard level. Stream-level metrics are for most common monitoring use cases in normal conditions. Shard-level metrics are for specific monitoring tasks, usually related to troubleshooting. For more information, see Monitoring Amazon Kinesis with Amazon CloudWatch in the *Amazon Kinesis Developer Guide*.

Contents

Basic Stream-level Metrics

The `AWS/Kinesis` namespace includes the following stream-level metrics.

Kinesis Data Streams sends these stream-level metrics to CloudWatch every minute. These metrics are always available.

Metric	Description
`GetRecords.Bytes`	The number of bytes retrieved from the Kinesis stream, measured over the specified time period. Minimum, Maximum, and Average statistics represent the bytes in a single `GetRecords` operation for the stream in the specified time period.
	Shard-level metric name: `OutgoingBytes`
	Dimensions: StreamName
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Bytes
`GetRecords.IteratorAge`	This metric is deprecated. Use `GetRecords.IteratorAgeMilliseconds`.
`GetRecords.IteratorAgeMilliseconds`	The age of the last record in all `GetRecords` calls made against an Kinesis stream, measured over the specified time period. Age is the difference between the current time and when the last record of the `GetRecords` call was written to the stream. The Minimum and Maximum statistics can be used to track the progress of Kinesis consumer applications. A value of zero indicates that the records being read are completely caught up with the stream.
	Shard-level metric name: `IteratorAgeMilliseconds`
	Dimensions: StreamName
	Statistics: Minimum, Maximum, Average, Samples
	Units: Milliseconds
`GetRecords.Latency`	The time taken per `GetRecords` operation, measured over the specified time period.
	Dimensions: StreamName
	Statistics: Minimum, Maximum, Average
	Units: Milliseconds
`GetRecords.Records`	The number of records retrieved from the shard, measured over the specified time period. Minimum, Maximum, and Average statistics represent the records in a single `GetRecords` operation for the stream in the specified time period.
	Shard-level metric name: `OutgoingRecords`
	Dimensions: StreamName
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Count

Metric	Description
GetRecords.Success	The number of successful `GetRecords` operations per stream, measured over the specified time period. Dimensions: StreamName Statistics: Average, Sum, Samples Units: Count
IncomingBytes	The number of bytes successfully put to the Kinesis stream over the specified time period. This metric includes bytes from `PutRecord` and `PutRecords` operations. Minimum, Maximum, and Average statistics represent the bytes in a single put operation for the stream in the specified time period. Shard-level metric name: `IncomingBytes` Dimensions: StreamName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Bytes
IncomingRecords	The number of records successfully put to the Kinesis stream over the specified time period. This metric includes record counts from `PutRecord` and `PutRecords` operations. Minimum, Maximum, and Average statistics represent the records in a single put operation for the stream in the specified time period. Shard-level metric name: `IncomingRecords` Dimensions: StreamName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count
PutRecord.Bytes	The number of bytes put to the Kinesis stream using the `PutRecord` operation over the specified time period. Dimensions: StreamName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Bytes
PutRecord.Latency	The time taken per `PutRecord` operation, measured over the specified time period. Dimensions: StreamName Statistics: Minimum, Maximum, Average Units: Milliseconds

Metric	Description
PutRecord.Success	The number of successful `PutRecord` operations per Kinesis stream, measured over the specified time period. Average reflects the percentage of successful writes to a stream. Dimensions: StreamName Statistics: Average, Sum, Samples Units: Count
PutRecords.Bytes	The number of bytes put to the Kinesis stream using the `PutRecords` operation over the specified time period. Dimensions: StreamName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Bytes
PutRecords.Latency	The time taken per `PutRecords` operation, measured over the specified time period. Dimensions: StreamName Statistics: Minimum, Maximum, Average Units: Milliseconds
PutRecords.Records	The number of successful records in a `PutRecords` operation per Kinesis stream, measured over the specified time period. Dimensions: StreamName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count
PutRecords.Success	The number of `PutRecords` operations where at least one record succeeded, per Kinesis stream, measured over the specified time period. Dimensions: StreamName Statistics: Average, Sum, Samples Units: Count

Metric	Description
`ReadProvisionedThroughputExceeded`	The number of `GetRecords` calls throttled for the stream over the specified time period. The most commonly used statistic for this metric is Average.
	When the Minimum statistic has a value of 1, all records were throttled for the stream during the specified time period.
	When the Maximum statistic has a value of 0 (zero), no records were throttled for the stream during the specified time period.
	Shard-level metric name: `ReadProvisionedThroughputExceeded`
	Dimensions: StreamName
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Count
`SubscribeToShard.RateExceeded`	This metric is emitted when a new subscription attempt fails because there is already an active subscription by the same consumer or you exceed the number of calls per second allowed for this operation.
	Dimensions: StreamName, ConsumerName
`SubscribeToShard.Success`	This metric records whether the `SubscribeToShard` subscription was successfully established. The subscription only lives for at most 5 minutes. Therefore, this metric is emitted at least once every 5 minutes.
	Dimensions: StreamName, ConsumerName
`SubscribeToShardEvent.Bytes`	The number of bytes received from the shard, measured over the specified time period. Minimum, Maximum, and Average statistics represent the bytes published in a single event for the specified time period.
	Shard-level metric name: `OutgoingBytes`
	Dimensions: StreamName, ConsumerName
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Bytes
`SubscribeToShardEvent.MillisBehindLatest`	The difference between the current time and when the last record of the `SubscribeToShard` event was written to the stream.
	Dimensions: StreamName, ConsumerName
	Statistics: Minimum, Maximum, Average, Samples
	Units: Milliseconds

Metric	Description
`SubscribeToShardEvent.Records`	The number of records received from the shard, measured over the specified time period. Minimum, Maximum, and Average statistics represent the records in a single event for the specified time period. Shard-level metric name: `OutgoingRecords` Dimensions: StreamName, ConsumerName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count
`SubscribeToShardEvent.Success`	This metric is emitted every time an event is published successfully. It is emitted only when there is an active subscription. Dimensions: StreamName, ConsumerName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count
`WriteProvisionedThroughputExceeded`	The number of records rejected due to throttling for the stream over the specified time period. This metric includes throttling from `PutRecord` and `PutRecords` operations. The most commonly used statistic for this metric is Average. When the Minimum statistic has a non-zero value, records were being throttled for the stream during the specified time period. When the Maximum statistic has a value of 0 (zero), no records were being throttled for the stream during the specified time period. Shard-level metric name: `WriteProvisionedThroughputExceeded` Dimensions: StreamName Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count

Enhanced Shard-level Metrics

The `AWS/Kinesis` namespace includes the following shard-level metrics.

Kinesis sends the following shard-level metrics to CloudWatch every minute. These metrics are not enabled by default. There is a charge for enhanced metrics emitted from Kinesis. For more information, see Amazon CloudWatch Pricing under the heading *Amazon CloudWatch Custom Metrics*. The charges are given per shard per metric per month.

Metric	Description
`IncomingBytes`	The number of bytes successfully put to the shard over the specified time period. This metric includes bytes from `PutRecord` and `PutRecords` operations. Minimum, Maximum, and Average statistics represent the bytes in a single put operation for the shard in the specified time period. Stream-level metric name: `IncomingBytes` Dimensions: StreamName, ShardId Statistics: Minimum, Maximum, Average, Sum, Samples Units: Bytes
`IncomingRecords`	The number of records successfully put to the shard over the specified time period. This metric includes record counts from `PutRecord` and `PutRecords` operations. Minimum, Maximum, and Average statistics represent the records in a single put operation for the shard in the specified time period. Stream-level metric name: `IncomingRecords` Dimensions: StreamName, ShardId Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count
`IteratorAgeMilliseconds`	The age of the last record in all `GetRecords` calls made against a shard, measured over the specified time period. Age is the difference between the current time and when the last record of the `GetRecords` call was written to the stream. The Minimum and Maximum statistics can be used to track the progress of Kinesis consumer applications. A value of 0 (zero) indicates that the records being read are completely caught up with the stream. Stream-level metric name: `GetRecords.IteratorAgeMilliseconds` Dimensions: StreamName, ShardId Statistics: Minimum, Maximum, Average, Samples Units: Milliseconds
`OutgoingBytes`	The number of bytes retrieved from the shard, measured over the specified time period. Minimum, Maximum, and Average statistics represent the bytes returned in a single `GetRecords` operation or published in a single `SubscribeToShard` event for the shard in the specified time period. Stream-level metric name: `GetRecords.Bytes`

Metric	Description
	Dimensions: StreamName, ShardId
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Bytes
`OutgoingRecords`	The number of records retrieved from the shard, measured over the specified time period. Minimum, Maximum, and Average statistics represent the records returned in a single `GetRecords` operation or published in a single `SubscribeToShard` event for the shard in the specified time period.
	Stream-level metric name: `GetRecords.Records`
	Dimensions: StreamName, ShardId
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Count
`ReadProvisionedThroughputExceeded`	The number of `GetRecords` calls throttled for the shard over the specified time period. This exception count covers all dimensions of the following limits: 5 reads per shard per second or 2 MB per second per shard. The most commonly used statistic for this metric is Average.
	When the Minimum statistic has a value of 1, all records were throttled for the shard during the specified time period.
	When the Maximum statistic has a value of 0 (zero), no records were throttled for the shard during the specified time period.
	Stream-level metric name: `ReadProvisionedThroughputExceeded`
	Dimensions: StreamName, ShardId
	Statistics: Minimum, Maximum, Average, Sum, Samples
	Units: Count

Metric	Description
WriteProvisionedThroughputExceeded	The number of records rejected due to throttling for the shard over the specified time period. This metric includes throttling from `PutRecord` and `PutRecords` operations and covers all dimensions of the following limits: 1,000 records per second per shard or 1 MB per second per shard. The most commonly used statistic for this metric is Average. When the Minimum statistic has a non-zero value, records were being throttled for the shard during the specified time period. When the Maximum statistic has a value of 0 (zero), no records were being throttled for the shard during the specified time period. Stream-level metric name: `WriteProvisionedThroughputExceeded` Dimensions: StreamName, ShardId Statistics: Minimum, Maximum, Average, Sum, Samples Units: Count

Dimensions for Amazon Kinesis Metrics

You can use the following dimensions to filter the metrics for Amazon Kinesis Data Streams.

Dimension	Description
StreamName	The name of the Kinesis stream.
ShardId	The shard ID within the Kinesis stream.

Amazon Kinesis Video Streams Metrics and Dimensions

Metrics

The `AWS/KinesisVideo` namespace includes the following metrics.

Metric	Description
PutMedia.Requests	Number of `PutMedia` API requests for a given stream. Units: Count
PutMedia.IncomingBytes	Number of bytes received as part of `PutMedia` for the stream.

Metric	Description
	Units: Bytes
`PutMedia.IncomingFragments`	Number of complete fragments received as part of `PutMedia` for the stream. Units: Count
`PutMedia.IncomingFrames`	Number of complete frames received as part of `PutMedia` for the stream. Units: Count
`PutMedia.ActiveConnections`	The total number of connections to the service host. Units: Count
`PutMedia.ConnectionErrors`	Errors while establishing `PutMedia` connection for the stream. Units: Count
`PutMedia.FragmentIngestionLatency`	Time difference between when the first and last bytes of a fragment are received by Kinesis Video Streams. Units: Milliseconds
`PutMedia.FragmentPersistLatency`	Time taken from when the complete fragment data is received and archived. Units: Count
`PutMedia.Latency`	Time difference between the request and the HTTP response from InletService while establishing the connection. Units: Count
`PutMedia.BufferingAckLatency`	Time difference between when the first byte of a new fragment is received by Kinesis Video Streams and when the Buffering ACK is sent for the fragment. Units: Milliseconds
`PutMedia.ReceivedAckLatency`	Time difference between when the last byte of a new fragment is received by Kinesis Video Streams and when the Received ACK is sent for the fragment. Units: Milliseconds
`PutMedia.PersistedAckLatency`	Time difference between when the last byte of a new fragment is received by Kinesis Video Streams and when the Persisted ACK is sent for the fragment. Units: Milliseconds
`PutMedia.ErrorAckCount`	Number of Error ACKs sent while doing `PutMedia` for the stream. Units: Count

Metric	Description
`PutMedia.Success`	1 for each fragment successfully written; 0 for every failed fragment. The average value of this metric indicates how many complete, valid fragments are sent. Units: Count
`GetMedia.Requests`	Number of `GetMedia` API requests for a given stream. Units: Count
`GetMedia.OutgoingBytes`	Total number of bytes sent out from the service as part of the `GetMedia` API for a given stream. Units: Bytes
`GetMedia.OutgoingFragments`	Number of fragments sent while doing `GetMedia` for the stream. Units: Count
`GetMedia.OutgoingFrames`	Number of frames sent during `GetMedia` on the given stream. Units: Count
`GetMedia.MillisBehindNow`	Time difference between the current server time stamp and the server time stamp of the last fragment sent. Units: Milliseconds
`GetMedia.ConnectionErrors`	The number of connections that were not successfully established. Units: Count
`GetMedia.Success`	1 for every fragment successfully sent; 0 for every failure. The average value indicates the rate of success. Units: Count
`GetMediaForFragmentList.OutgoingBytes`	Total number of bytes sent out from the service as part of the `GetMediaForFragmentList` API for a given stream. Units: Bytes
`GetMediaForFragmentList.OutgoingFragments`	Total number of fragments sent out from the service as part of the `GetMediaForFragmentList` API for a given stream. Units: Count
`GetMediaForFragmentList.OutgoingFrames`	Total number of frames sent out from the service as part of the `GetMediaForFragmentList` API for a given stream. Units: Count

Metric	Description
GetMediaForFragmentList.Requests	Number of GetMediaForFragmentList API requests for a given stream. Units: Count
GetMediaForFragmentList.Success	1 for every fragment successfully sent; 0 for every failure. The average value indicates the rate of success. Units: Count
ListFragments.Latency	Latency of the ListFragments API calls for the given stream name. Units: Milliseconds
GetHLSStreamingSessionURL.Latency	Latency of the GetHLSStreamingSessionURL API calls for the given stream name. Units: Milliseconds
GetHLSStreamingSessionURL.Requests	Number of GetHLSStreamingSessionURL API requests for a given stream. Units: Count
GetHLSStreamingSessionURL.Success	1 for every successful request; 0 for every failure. The average value indicates the rate of success. Units: Count
GetHLSMasterPlaylist.Latency	Latency of the GetHLSMasterPlaylist API calls for the given stream name. Units: Milliseconds
GetHLSMasterPlaylist.Requests	Number of GetHLSMasterPlaylist API requests for a given stream. Units: Count
GetHLSMasterPlaylist.Success	1 for every successful request; 0 for every failure. The average value indicates the rate of success. Units: Count
GetHLSMediaPlaylist.Latency	Latency of the GetHLSMediaPlaylist API calls for the given stream name. Units: Milliseconds
GetHLSMediaPlaylist.Requests	Number of GetHLSMediaPlaylist API requests for a given stream. Units: Count
GetHLSMediaPlaylist.Success	1 for every successful request; 0 for every failure. The average value indicates the rate of success. Units: Count

Metric	Description
`GetMP4InitFragment.Latency`	Latency of the `GetMP4InitFragment` API calls for the given stream name. Units: Milliseconds
`GetMP4InitFragment.Requests`	Number of `GetMP4InitFragment` API requests for a given stream. Units: Count
`GetMP4InitFragment.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. Units: Count
`GetMP4MediaFragment.Latency`	Latency of the `GetMP4MediaFragment` API calls for the given stream name. Units: Milliseconds
`GetMP4MediaFragment.Requests`	Number of `GetMP4MediaFragment` API requests for a given stream. Units: Count
`GetMP4MediaFragment.Success`	1 for every successful request; 0 for every failure. The average value indicates the rate of success. Units: Count
`GetMP4MediaFragment.OutgoingBytes`	Total number of bytes sent out from the service as part of the `GetMP4MediaFragment` API for a given stream. Units: Bytes

Dimensions for Amazon Kinesis Video Streams Metrics

You can use the following dimensions to filter the metrics for Amazon Kinesis Video Streams.

Dimension	Description
`StreamName`	The name of the Kinesis video stream.

AWS Key Management Service Metrics and Dimensions

When you use AWS Key Management Service (AWS KMS) to import key material into a customer master key (CMK) and set it to expire, AWS KMS sends metrics and dimensions to CloudWatch. For more information, see Monitoring with Amazon CloudWatch in the *AWS Key Management Service Developer Guide*.

AWS KMS Metrics

The `AWS/KMS` namespace includes the following metrics.

SecondsUntilKeyMaterialExpiration

This metric tracks the number of seconds remaining until imported key material expires. This metric is valid only for CMKs whose origin is `EXTERNAL` and whose key material is or was set to expire. The most useful statistic for this metric is `Minimum`, which tells you the smallest amount of time remaining for all data points in the specified statistic period. The only valid unit for this metric is `Seconds`.

Use this metric to track the amount of time that remains until your imported key material expires. When that amount of time falls below a threshold that you define, you might want to take action such as reimporting the key material with a new expiration date. You can create a CloudWatch alarm to notify you when that happens. For more information, see Creating CloudWatch Alarms to Monitor AWS KMS Metrics in the *AWS Key Management Service Developer Guide*.

Dimensions for AWS KMS Metrics

AWS KMS metrics use the `AWS/KMS` namespace and have only one valid dimension: `KeyId`. You can use this dimension to view metric data for a specific CMK or set of CMKs.

AWS Lambda Metrics and Dimensions

AWS Lambda sends metrics to CloudWatch every minute. For more information, see Troubleshooting and Monitoring AWS Lambda Functions with Amazon CloudWatch in the *AWS Lambda Developer Guide*.

AWS Lambda CloudWatch Metrics

The `AWS/Lambda` namespace includes the following metrics.

Metric	Description
`Invocations`	Measures the number of times a function is invoked in response to an event or invocation API call. This replaces the deprecated RequestCount metric. This includes successful and failed invocations, but does not include throttled attempts. This equals the billed requests for the function. Note that AWS Lambda only sends these metrics to CloudWatch if they have a nonzero value. Units: Count
`Errors`	Measures the number of invocations that failed due to errors in the function (response code 4XX). This replaces the deprecated ErrorCount metric. Failed invocations may trigger a retry attempt that succeeds. This includes: • Handled exceptions (for example, context.fail(error)) • Unhandled exceptions causing the code to exit • Out of memory exceptions • Timeouts • Permissions errors

Metric	Description
	This does not include invocations that fail due to invocation rates exceeding default concurrent limits (error code 429) or failures due to internal service errors (error code 500). Units: Count
DeadLetterErrors	Incremented when Lambda is unable to write the failed event payload to your configured Dead Letter Queues. This could be due to the following: • Permissions errors • Throttles from downstream services • Misconfigured resources • Timeouts Units: Count
Duration	Measures the elapsed wall clock time from when the function code starts executing as a result of an invocation to when it stops executing. This replaces the deprecated Latency metric. The maximum data point value possible is the function timeout configuration. The billed duration will be rounded up to the nearest 100 millisecond. Note that AWS Lambda only sends these metrics to CloudWatch if they have a nonzero value. Units: Milliseconds
Throttles	Measures the number of Lambda function invocation attempts that were throttled due to invocation rates exceeding the customer's concurrent limits (error code 429). Failed invocations may trigger a retry attempt that succeeds. Units: Count
IteratorAge	Emitted for stream-based invocations only (functions triggered by an Amazon DynamoDB stream or Kinesis stream). Measures the age of the last record for each batch of records processed. Age is the difference between the time Lambda received the batch, and the time the last record in the batch was written to the stream. Units: Milliseconds
ConcurrentExecutions	Emitted as an aggregate metric for all functions in the account, and for functions that have a custom concurrency limit specified. Not applicable for versions or aliases. Measures the sum of concurrent executions for a given function at a given point in time. Must be viewed as an average metric if aggregated across a time period. Units: Count
UnreservedConcurrentExecutions	Emitted as an aggregate metric for all functions in the account only. Not applicable for functions, versions, or aliases. Represents the sum of the concurrency of the functions that do not have a custom concurrency limit specified. Must be viewed as an average metric if aggregated across a time period. Units: Count

Errors/Invocations Ratio
When calculating the error rate on Lambda function invocations, it's important to distinguish between an invocation request and an actual invocation. It is possible for the error rate to exceed the number of billed Lambda function invocations. Lambda reports an invocation metric only if the Lambda function code is executed. If the invocation request yields a throttling or other initialization error that prevents the Lambda function code from being invoked, Lambda will report an error, but it does not log an invocation metric.

- Lambda emits `Invocations=1` when the function is executed. If the Lambda function is not executed, nothing is emitted.
- Lambda emits a data point for `Errors` for each invoke request. `Errors=0` means that there is no function execution error. `Errors=1` means that there is a function execution error.
- Lambda emits a data point for `Throttles` for each invoke request. `Throttles=0` means there is no invocation throttle. `Throttles=1` means there is an invocation throttle.

Dimensions for AWS Lambda Metrics

Lambda data can be filtered along any of the following dimensions in the table below.

AWS Lambda CloudWatch Dimensions

You can use the dimensions in the following table to refine the metrics returned for your Lambda functions.

Dimension	Description
FunctionName	Filters the metric data by Lambda function.
Resource	Filters the metric data by Lambda function resource, such as function version or alias.
ExecutedVersion	Filters the metric data by Lambda function versions. This only applies to alias invocations.

Amazon Lex Metrics

The following tables describe the Amazon Lex metrics.

CloudWatch Metrics for Amazon Lex Runtime

The following table describes the Amazon Lex runtime metrics.

Metric	Description
RuntimeInvalidLambdaResponses	The number of invalid Lambda responses in the specified period. Valid dimension for the `PostContent` operation with the `Text` or `Speech InputMode`: - `BotName, BotAlias, Operation, InputMode` Valid dimension for the `PostText` operation:

Metric	Description
	• `BotName, BotAlias, Operation`
`RuntimeLambdaErrors`	The number of AWS Lambda runtime errors in the specified period. Valid dimension for the `PostContent` operation with the `Text` or `Speech InputMode`: • `BotName, BotAlias, Operation, InputMode` Valid dimension for the `PostText` operation: • `BotName, BotAlias, Operation`
`MissedUtteranceCount`	The number of utterances that were not recognized in the specified period. Valid dimensions for the `PostContent` operation with the `Text` or `Speech InputMode`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Valid dimensions for the `PostText` operation: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation`
`RuntimePollyErrors`	The number of invalid Amazon Polly responses in the specified period. Valid dimension for the `PostContent` operation with the `Text` or `Speech InputMode`: • `BotName, BotAlias, Operation, InputMode` Valid dimension for the `PostText` operation: • `BotName, BotAlias, Operation`
`RuntimeRequestCount`	The number of runtime requests in the specified period. Valid dimensions for the `PostContent` operation with the `Text` or `Speech InputMode`: • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Valid dimensions for the `PostText` operation: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unit: Count

Metric	Description
`RuntimeSucessfulRequestLatency`	The latency for successful requests between the time that the request was made and the response was passed back.

Valid dimensions for the `PostContent` operation with the `Text` or `Speech InputMode`:

- `BotName, BotVersion, Operation, InputMode`
- `BotName, BotAlias, Operation, InputMode`

Valid dimensions for the `PostText` operation:

- `BotName, BotVersion, Operation`
- `BotName, BotAlias, Operation`

Unit: Milliseconds |
| `RuntimeSystemErrors` | The number of system errors in the specified period. The response code range for a system error is 500 to 599.

Valid dimension for the `PostContent` operation with the `Text` or `Speech InputMode`:

- `BotName, BotAlias, Operation, InputMode`

Valid dimension for the `PostText` operation:

- `BotName, BotAlias, Operation`

Unit: Count |
| `RuntimeThrottledEvents` | The number of throttled requests. Amazon Lex throttles a request when it receives more requests than the limit of transactions per second set for your account. If the limit set for your account is frequently exceeded, you can request a limit increase. To request an increase, see AWS Service Limits.

Valid dimension for the `PostContent` operation with the `Text` or `Speech InputMode`:

- BotName, BotAlias, Operation, InputMode

Valid dimension for the `PostText` operation:

- `BotName, BotAlias, Operation`

Unit: Count |

Metric	Description
`RuntimeUserErrors`	The number of user errors in the specified period. The response code range for a user error is 400 to 499. Valid dimension for the `PostContent` operation with `Text` or `Speech` `InputMode`: • `BotName, BotAlias, Operation, InputMode` Valid dimension for the `PostText` operation: • `BotName, BotAlias, Operation` Unit: Count

Amazon Lex runtime metrics use the `AWS/Lex` namespace, and provide metrics in the following dimensions. You can group metrics by dimenstions in the CloudWatch console:

Dimension	Description
`BotName, BotAlias, Operation, InputMode`	Groups metrics by the bot's alias, the bot's name, the operation (`PostContent`), and by whether the input was text or speech.
`BotName, BotVersion, Operation, InputMode`	Groups metrics by the bot's name, the version of the bot, the operation (`PostContent`), and by whether the input was text or speech.
`BotName, BotVersion, Operation`	Groups metrics by the bot's name, the bots version, and by the operation, `PostText`.
`BotName, BotAlias, Operation`	Groups metrics by the bot's name, the bot's alias, and by the operation, `PostText`.

CloudWatch Metrics for Amazon Lex Channel Associations

A channel association is the association between Amazon Lex and a messaging channel, such as Facebook. The following table describes the Amazon Lex channel association metrics.

Metric	Description
`BotChannelAuthErrors`	The number of authentication errors returned by the messaging channel in the specified time period. An authentication error indicates that the secret token provided during channel creation is invalid or has expired.
`BotChannelConfigurationErrors`	The number of configuration errors in the specified period. A configuration error indicates that one or more configuration entries for the channel are invalid.
`BotChannelInboundThrottledEvents`	The number of times that messages that were sent by the messaging channel were throttled by Amazon Lex in the specified period.

Metric	Description
`BotChannelOutboundThrottledEvents`	The number of times that outbound events from Amazon Lex to the messaging channel were throttled in the specified time period.
`BotChannelRequestCount`	The number of requests made on a channel in the specified time period.
`BotChannelResponseCardErrors`	The number of times that Amazon Lex could not post response cards in the specified period.
`BotChannelSystemErrors`	The number of internal errors that occurred in Amazon Lex for a channel in the specified period.

Amazon Lex channel association metrics use the `AWS/Lex` namespace, and provide metrics for the following dimension. You can group metrics by dimenstions in the CloudWatch console::

Dimension	Description
`BotAlias, BotChannelName, BotName, Source`	Group metrics by the bot's alias, the channel name, the bot's name, and by the source of traffic.

Amazon Machine Learning Metrics and Dimensions

Amazon Machine Learning sends metrics to CloudWatch every five minutes. For more information, see Monitoring Amazon ML with Amazon CloudWatch Metrics in the *Amazon Machine Learning Developer Guide*.

Amazon ML Metrics

The `AWS/ML` namespace includes the following metrics.

Metric	Description
`PredictCount`	The number of observations received by Amazon ML, measured over the specified time period. Units: Count
`PredictFailureCount`	The number of invalid or malformed observations received by Amazon ML, measured over the specified time period. Units: Count

Dimensions for Amazon Machine Learning Metrics

Amazon ML data can be filtered along any of the following dimensions in the table below.

Dimension	Description
`MLModelId`	The identifier of an Amazon ML model. All available statistics are filtered by `MLModelId`.

Dimension	Description
RequestMode	An indicator specifying whether observations were received as part of a batch prediction request or as real-time predict requests. All available statistics are filtered by RequestMode.

Amazon MQ Metrics

Amazon MQ and Amazon CloudWatch are integrated so you can use CloudWatch to view and analyze metrics for your ActiveMQ broker and the broker's destinations (queues and topics). You can view and analyze your Amazon MQ metrics from the CloudWatch console, the AWS CLI, or the CloudWatch CLI. CloudWatch metrics for Amazon MQ are automatically polled from the broker and then pushed to CloudWatch every minute. For more information, see Monitoring Amazon MQ with CloudWatch in the *Amazon MQ Developer Guide*.

> **Note**
> The following statistics are valid for all of the metrics:
>
> - Average
> - Minimum
> - Maximum
> - Sum

The AWS/AmazonMQ namespace includes the following metrics.

Broker Metrics

Metric	Unit	Description
CpuCreditBalance	Credits (vCPU-minutes)	**Important** This metric is available only for the mq.t2.micro broker instance type. CPU credit metrics are available only at five-minute intervals. The number of earned CPU credits that an instance has accrued since it was launched or started (including the number of launch credits). The credit balance is available for the broker instance to spend on bursts beyond the baseline CPU utilization. Credits are accrued in the credit balance after they're earned and removed from the credit balance after they're spent. The credit balance has a maximum limit. Once the limit is reached,

Metric	Unit	Description
		any newly earned credits are discarded.
`CpuUtilization`	Percent	The percentage of allocated EC2 compute units that the broker currently uses.
`CurrentConnectionsCount`	Count	The current number of active connections on the current broker.
`HeapUsage`	Percent	The percentage of the ActiveMQ JVM memory limit that the broker currently uses.
`NetworkIn`	Bytes	The volume of incoming traffic for the broker.
`NetworkOut`	Bytes	The volume of outgoing traffic for the broker.
`TotalConsumerCount`	Count	The number of message consumers subscribed to destinations on the current broker.
`TotalMessageCount`	Count	The number of messages stored on the broker.
`TotalProducerCount`	Count	The number of message producers active on destinations on the current broker.

Dimension for Broker Metrics

Dimension	Description
`Broker`	The name of the broker. **Note** A single-instance broker has the suffix `-1`. An active-standby broker for high availability has the suffixes `-1` and `-2` for its redundant pair.

Destination (Queue and Topic) Metrics

Important
The following metrics record only values since CloudWatch polled the metrics last:

- `EnqueueCount`
- `ExpiredCount`
- `DequeueCount`
- `DispatchCount`

Metric	Unit	Description
ConsumerCount	Count	The number of consumers subscribed to the destination.
EnqueueCount	Count	The number of messages sent to the destination.
EnqueueTime	Time (milliseconds)	The amount of time it takes the broker to accept a message from the producer and send it to the destination.
ExpiredCount	Count	The number of messages that couldn't be delivered because they expired.
DispatchCount	Count	The number of messages sent to consumers.
DequeueCount	Count	The number of messages acknowledged by consumers.
MemoryUsage	Percent	The percentage of the memory limit that the destination currently uses.
ProducerCount	Count	The number of producers for the destination.
QueueSize	Count	The number of messages in the queue. **Important** This metric applies only to queues.

Dimensions for Destination (Queue and Topic) Metrics

Dimension	Description
Broker	The name of the broker. **Note** A single-instance broker has the suffix -1. An active-standby broker for high availability has the suffixes -1 and -2 for its redundant pair.
Topic or Queue	The name of the topic or queue.

Amazon Neptune Metrics and Dimensions

Amazon Neptune sends metrics to CloudWatch. The namespace is AWS/Neptune. For more information, see Monitoring Neptune with CloudWatch in the *Neptune User Guide*

Neptune Metrics

The following metrics are available from Amazon Neptune. Neptune sends metrics to CloudWatch only when they have a non-zero value.

> **Note**
> For all Neptune metrics, the aggregation granularity is five minutes.

Metric	Description
CPUUtilization	The percentage of CPU utilization.
ClusterReplicaLag	For a read replica, the amount of lag when replicating updates from the primary instance, in milliseconds.
ClusterReplicaLagMaximum	The maximum amount of lag between the primary instance and each Neptune DB instance in the DB cluster, in milliseconds.
ClusterReplicaLagMinimum	The minimum amount of lag between the primary instance and each Neptune DB instance in the DB cluster, in milliseconds.
EngineUptime	The amount of time that the instance has been running, in seconds.
FreeableMemory	The amount of available random access memory, in bytes.
FreeLocaStorage	The amount of storage available for temporary tables and logs, in bytes. This metric reports the amount of storage available to each DB instance for temporary tables and logs. This value depends on the DB instance class (for pricing information, see the Amazon Neptune pricing page). You can increase the amount of free storage space for an instance by choosing a larger DB instance class for your instance.
GremlinErrors	Number of errors in Gremlin traversals.
GremlinRequests	Number of requests to Gremlin engine.
GremlinRequestsPerSec	Number of requests to Gremlin engine per second.
Http100	Number of HTTP 100 errors for the endpoint.
Http101	Number of HTTP 101 errors for the endpoint.
Http200	Number of HTTP 200 errors for the endpoint.
Http400	Number of HTTP 400 errors for the endpoint.
Http403	Number of HTTP 403 errors for the endpoint.
Http405	Number of HTTP 405 errors for the endpoint.

Metric	Description
Http413	Number of HTTP 413 errors for the endpoint.
Http429	Number of HTTP 429 errors for the endpoint.
Http500	Number of HTTP 500 errors for the endpoint.
Http501	Number of HTTP 501 errors for the endpoint.
LoaderErrors	Number of errors from Loader requests.
LoaderRequests	Number of Loader Requests.
NetworkReceiveThroughput	The incoming (Receive) network traffic on the DB instance, including both customer database traffic and Neptune traffic used for monitoring and replication, in bytes/second.
NetworkThroughput	The amount of network throughput both received from and transmitted to clients by each instance in the Neptune DB cluster, in bytes per second. This throughput does not include network traffic between instances in the DB cluster and the cluster volume.
NetworkTransmitThroughput	The outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Neptune traffic used for monitoring and replication, in bytes/second.
SparqlErrors	Number of errors in the SPARQL queries.
SparqlRequests	Number of requests to the SPARQL engine.
SparqlRequestsPerSec	Number of requests to the SPARQL engine per second.
StatusErrors	Number of errors from the status endpoint.
StatusRequests	Number of requests to the status endpoint.
VolumeBytesUsed	The amount of storage used by your Neptune DB instance, in bytes. This value affects the cost of the Neptune DB cluster.
VolumeReadIOPs	The average number of billed read I/O operations from a cluster volume, reported at 5-minute intervals. Billed read operations are calculated at the cluster volume level, aggregated from all instances in the Neptune DB cluster, and then reported at 5-minute intervals.
VolumeWriteIOPs	The average number of write disk I/O operations to the cluster volume, reported at 5-minute intervals.

Neptune Dimensions

The metrics for Neptune are qualified by the values for the account, graph name, or operation. You can use the CloudWatch console to retrieve Neptune data along with any of the dimensions in the following table.

Dimension	Description
`DBClusterIdentifier`	Filters the data you request for a specific database instance within a cluster.
`DBClusterIdentifier, Role`	Filters the data you request for a specific Neptune DB cluster, aggregating the metric by instance role (WRITER/READER). For example, you can aggregate metrics for all READER instances that belong to a cluster.
`DatabaseClass`	Filters the data you request for all instances in a database class. For example, you can aggregate metrics for all instances that belong to the database class db.r4.large
`DBClusterIdentifier, EngineName`	Filters the data by the cluster. The engine name for all Neptune instances is `neptune`.
`EngineName`	The engine name for all Neptune instances is `neptune`.

AWS OpsWorks Metrics and Dimensions

AWS OpsWorks sends metrics to CloudWatch for each active stack every minute. Detailed monitoring is enabled by default. For more information, see Monitoring in the *AWS OpsWorks User Guide*.

AWS OpsWorks Stacks Metrics

AWS OpsWorks Stacks sends the following metrics to CloudWatch every five minutes.

CPU Metrics

Metric	Description
`cpu_idle`	The percentage of time that the CPU is idle. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: `Average`, `Minimum`, `Maximum`, `Sum`, or `Data Samples`. Unit: None
`cpu_nice`	The percentage of time that the CPU is handling processes with a positive `nice` value, which have a lower scheduling priority. For more information about what this measures, see nice (Unix).

Metric	Description
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: `Average`, `Minimum`, `Maximum`, `Sum`, or `Data Samples`.
	Unit: None
cpu_steal	As AWS allocates hypervisor CPU resources among increasing numbers of instances, virtualization load rises, and can affect how often the hypervisor can perform requested work on an instance. `cpu_steal` measures the percentage of time that an instance is waiting for the hypervisor to allocate physical CPU resources.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: `Average`, `Minimum`, `Maximum`, `Sum`, or `Data Samples`.
	Unit: None
cpu_system	The percentage of time that the CPU is handling system operations.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: `Average`, `Minimum`, `Maximum`, `Sum`, or `Data Samples`.
	Unit: None
cpu_user	The percentage of time that the CPU is handling user operations.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: `Average`, `Minimum`, `Maximum`, `Sum`, or `Data Samples`.
	Unit: None

Metric	Description
cpu_waitio	The percentage of time that the CPU is waiting for input/output operations. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples. Unit: None

Memory Metrics

Metric	Description
memory_buffers	The amount of buffered memory. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples. Unit: None
memory_cached	The amount of cached memory. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples. Unit: None
memory_free	The amount of free memory. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples. Unit: None
memory_swap	The amount of swap space. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples.

Metric	Description
	Unit: None
memory_total	The total amount of memory.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples.
	Unit: None
memory_used	The amount of memory in use.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples.
	Unit: None

Load Metrics

Metric	Description
load_1	The load averaged over a one-minute window.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples.
	Unit: None
load_5	The load averaged over a five-minute window.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples.
	Unit: None
load_15	The load averaged over a 15-minute window.
	Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId.
	Valid Statistics: Average, Minimum, Maximum, Sum, or Data Samples.

Metric	Description
	Unit: None

Process Metrics

Metric	Description
`procs`	The number of active processes. Valid Dimensions: The IDs of the individual resources for which you are viewing metrics: StackId, LayerId, or InstanceId. Valid Statistics: `Average, Minimum, Maximum, Sum,` or `Data Samples.` Unit: None

Dimensions for AWS OpsWorks Metrics

AWS OpsWorks data can be filtered along any of the following dimensions in the table below.

Dimension	Description
`StackId`	Average values for a stack.
`LayerId`	Average values for a layer.
`InstanceId`	Average values for an instance.

Amazon Polly Metrics

Amazon Polly sends metrics to CloudWatch. For more information, see the *Amazon Polly Developer Guide*.

Amazon Polly Metrics

Amazon Polly produces the following metrics for each request. These metrics are aggregated and in one minute intervals sent to CloudWatch where they are available in the `AWS/Polly` namespace.

Metric	Description
`RequestCharacters`	The number of characters in the request. This is billable characters only and does not include SSML tags. Valid Dimension: Operation Valid Statistics: Minimum, Maximum, Average, SampleCount, Sum Unit: Count
`ResponseLatency`	The latency between when the request was made and the start of the streaming response.

Metric	Description
	Valid Dimensions: Operation
	Valid Statistics: Minimum, Maximum, Average, SampleCount
	Unit: milliseconds
`2XXCount`	HTTP 200 level code returned upon a successful response.
	Valid Dimensions: Operation
	Valid Statistics: Average, SampleCount, Sum
	Unit: Count
`4XXCount`	HTTP 400 level error code returned upon an error. For each successful response, a zero (0) is emitted.
	Valid Dimensions: Operation
	Valid Statistics: Average, SampleCount, Sum
	Unit: Count
`5XXCount`	HTTP 500 level error code returned upon an error. For each successful response, a zero (0) is emitted.
	Valid Dimensions: Operation
	Valid Statistics: Average, SampleCount, Sum
	Unit: Count

Dimensions for Amazon Polly Metrics

Amazon Polly provides metrics for the following dimension.

Dimension	Description
`Operation`	Metrics are grouped by the API method they refer to. Possible values are `SynthesizeSpeech`, `PutLexicon`, `DescribeVoices`, etc.

Amazon Redshift Metrics and Dimensions

Amazon Redshift sends metrics to CloudWatch for each active cluster every minute. Detailed monitoring is enabled by default. For more information, see Monitoring Amazon Redshift Cluster Performance in the *Amazon Redshift Cluster Management Guide*.

Amazon Redshift Metrics

The `AWS/Redshift` namespace includes the following metrics.

Metric	Description
`CPUUtilization`	The percentage of CPU utilization. For clusters, this metric represents an aggregation of all nodes (leader and compute) CPU utilization values. Units: Percent Dimensions: `NodeID`, `ClusterIdentifier`
`DatabaseConnections`	The number of database connections to a cluster. Units: Count Dimensions: `ClusterIdentifier`
`HealthStatus`	Indicates the health of the cluster. Every minute the cluster connects to its database and performs a simple query. If it is able to perform this operation successfully, the cluster is considered healthy. Otherwise, the cluster is unhealthy. An unhealthy status can occur when the cluster database is under extremely heavy load or if there is a configuration problem with a database on the cluster. **Note** In Amazon CloudWatch this metric is reported as 1 or 0 whereas in the Amazon CloudWatch console, this metric is displayed with the words `HEALTHY` or `UNHEALTHY` for convenience. When this metric is displayed in the Amazon CloudWatch console, sampling averages are ignored and only `HEALTHY` or `UNHEALTHY` are displayed. In Amazon CloudWatch, values different than 1 and 0 may occur because of sampling issue. Any value below 1 for `HealthStatus` is reported as 0 (`UNHEALTHY`). Units: 1/0 (`HEALTHY`/`UNHEALTHY` in the Amazon CloudWatch console) Dimensions: `ClusterIdentifier`
`MaintenanceMode`	Indicates whether the cluster is in maintenance mode. **Note** In Amazon CloudWatch this metric is reported as 1 or 0 whereas in the Amazon CloudWatch console, this metric is displayed with the words `ON` or `OFF` for convenience. When this metric is displayed in the Amazon CloudWatch console, sampling averages are ignored and only `ON` or `OFF` are displayed. In Amazon CloudWatch, values different than 1 and 0 may occur because of sampling issues. Any value greater than 0 for `MaintenanceMode` is reported as 1 (`ON`). Units: 1/0 (`ON`/`OFF` in the Amazon CloudWatch console). Dimensions: `ClusterIdentifier`
`NetworkReceiveThroughput`	The rate at which the node or cluster receives data. Units: Bytes/seconds (MB/s in the Amazon CloudWatch console) Dimensions: `NodeID`, `ClusterIdentifier`

Metric	Description
NetworkTransmitThroughput	The rate at which the node or cluster writes data. Units: Bytes/second (MB/s in the Amazon CloudWatch console) Dimensions: `NodeID`, `ClusterIdentifier`
PercentageDiskSpaceUsed	The percent of disk space used. Units: Percent Dimensions: `NodeID`, `ClusterIdentifier`
QueriesCompletedPerSecond	This metric is used to determine Query Throughput. The metric is the average number of queries completed per second, reported in five-minute intervals. Units: Count/second Dimensions: latency
QueryDuration	The average amount of time to complete a query. Reported in five-minute intervals. Units: Microseconds Dimensions: latency
QueryRuntimeBreakdown	The amount of time all active queries have spent in various stages of execution during the previous five minutes. Units: Milliseconds Dimensions: Stage
ReadIOPS	The average number of disk read operations per second. Units: Count/second Dimensions: `NodeID`
ReadLatency	The average amount of time taken for disk read I/O operations. Units: Seconds Dimensions: `NodeID`
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes (GB/s in the Amazon CloudWatch console) Dimensions: `NodeID`
TotalTableCount	The number of user tables created at a particular point in time. This total doesn't include Amazon Redshift Spectrum tables. Units: Count Dimensions: `ClusterIdentifier`

Metric	Description
WLMQueriesCompletedPerSecond	This metric is used to determine Query Throughput for a Workload Management queue. The metric is the average number of queries completed per second for a Workload Management (WLM) queue, reported in five-minute intervals. Units: Count/second Dimensions: `wlmid`
WLMQueryDuration	The average length of time to complete a query for a Workload Management (WLM) queue. Reported in five-minute intervals. Units: Microseconds Dimensions: `wlmid`
WLMQueueLength	The number of queries in the queue for a Workload Management (WLM) queue. Units: Count Dimensions: `service class`
WriteIOPS	The average number of disk write operations per second. Units: Count/seconds Dimensions: `NodeID`
WriteLatency	The average amount of time taken for disk write I/O operations. Units: Seconds Dimensions: `NodeID`
WriteThroughput	The average number of bytes written to disk per second. Units: Bytes (GB/s in the Amazon CloudWatch console) Dimensions: `NodeID`

Dimensions for Amazon Redshift Metrics

Amazon Redshift data can be filtered along any of the following dimensions in the table below.

Amazon Redshift data can be filtered along any of the dimensions in the table following.

Dimension	Description
latency	Values are short, medium, and long. Short is less than 10 seconds, medium is between 10 seconds and 10 minutes, and long is over 10 minutes.
NodeID	Filters requested data that is specific to the nodes of a cluster. `NodeID` will be either "Leader", "Shared", or "Compute-N" where N is 0, 1, ... for the number of nodes in the cluster. "Shared"

Dimension	Description
	means that the cluster has only one node, i.e. the leader node and compute node are combined. Metrics are reported for the leader node and compute nodes only for `CPUUtilization`, `NetworkTransmitThroughput`, and `ReadIOPS`. Other metrics that use the `NodeId` dimension are reported only for compute nodes.
`ClusterIdentifier`	Filters requested data that is specific to the cluster. Metrics that are specific to clusters include `HealthStatus`, `MaintenanceMode`, and `DatabaseConnections`. In general metrics in for this dimension (e.g. `ReadIOPS`) that are also metrics of nodes represent an aggregate of the node metric data. You should take care in interpreting these metrics because they aggregate behavior of leader and compute nodes.
`service class`	The identifier for a WLM service class.
`Stage`	The execution stages for a query. The possible values are: • QueryPlanning: Time spent parsing and optimizing SQL statements . • QueryWaiting: Time spent waiting in the wlm queue. • QueryExecutingRead: Time spent executing read queries. • QueryExecutingInsert: Time spent executing insert queries. • QueryExecutingDelete: Time spent executing delete queries. • QueryExecutingUpdate: Time spent executing update queries. • QueryExecutingCtas: Time spent executing create table as ... queries. • QueryExecutingUnload: Time spent executing unload queries. • QueryExecutingCopy: Time spent executing copy queries. • QueryCommit: Time spent committing.
`wmlid`	The identifier for a Workload Management Queue.

Amazon RDS Metrics and Dimensions

Amazon Relational Database Service sends metrics to CloudWatch for each active database instance every minute. Detailed monitoring is enabled by default. For more information, see Monitoring a DB Instance in the *Amazon RDS User Guide*.

Amazon RDS Metrics

The `AWS/RDS` namespace includes the following metrics.

Metric	Description
`BinLogDiskUsage`	The amount of disk space occupied by binary logs on the master. Applies to MySQL read replicas. Units: Bytes

Metric	Description
BurstBalance	The percent of General Purpose SSD (gp2) burst-bucket I/O credits available. Units: Percent
CPUUtilization	The percentage of CPU utilization. Units: Percent
CPUCreditUsage	[T2 instances] The number of CPU credits spent by the instance for CPU utilization. One CPU credit equals one vCPU running at 100% utilization for one minute or an equivalent combination of vCPUs, utilization, and time (for example, one vCPU running at 50% utilization for two minutes or two vCPUs running at 25% utilization for two minutes). CPU credit metrics are available at a five-minute frequency only. If you specify a period greater than five minutes, use the Sum statistic instead of the Average statistic. Units: Credits (vCPU-minutes)
CPUCreditBalance	[T2 instances] The number of earned CPU credits that an instance has accrued since it was launched or started. For T2 Standard, the CPUCreditBalance also includes the number of launch credits that have been accrued. Credits are accrued in the credit balance after they are earned, and removed from the credit balance when they are spent. The credit balance has a maximum limit, determined by the instance size. Once the limit is reached, any new credits that are earned are discarded. For T2 Standard, launch credits do not count towards the limit. The credits in the CPUCreditBalance are available for the instance to spend to burst beyond its baseline CPU utilization. When an instance is running, credits in the CPUCreditBalance do not expire. When the instance stops, the CPUCreditBalance does not persist, and all accrued credits are lost. CPU credit metrics are available at a five-minute frequency only. Units: Credits (vCPU-minutes)
DatabaseConnections	The number of database connections in use. Units: Count
DiskQueueDepth	The number of outstanding IOs (read/write requests) waiting to access the disk. Units: Count
FreeableMemory	The amount of available random access memory. Units: Bytes

Metric	Description
FreeStorageSpace	The amount of available storage space. Units: Bytes
MaximumUsedTransactionIDs	The maximum transaction ID that has been used. Applies to PostgreSQL. Units: Count
NetworkReceiveThroughput	The incoming (Receive) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes/second
NetworkTransmitThroughput	The outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes/second
OldestReplicationSlotLag	The lagging size of the replica lagging the most in terms of WAL data received. Applies to PostgreSQL. Units: Megabytes
ReadIOPS	The average number of disk read I/O operations per second. Units: Count/Second
ReadLatency	The average amount of time taken per disk I/O operation. Units: Seconds
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes/Second
ReplicaLag	The amount of time a Read Replica DB instance lags behind the source DB instance. Applies to MySQL, MariaDB, and PostgreSQL Read Replicas. Units: Seconds
ReplicationSlotDiskUsage	The disk space used by replication slot files. Applies to PostgreSQL. Units: Megabytes
SwapUsage	The amount of swap space used on the DB instance. This metric is not available for SQL Server. Units: Bytes
TransactionLogsDiskUsage	The disk space used by transaction logs. Applies to PostgreSQL. Units: Megabytes

Metric	Description
`TransactionLogsGeneration`	The size of transaction logs generated per second. Applies to PostgreSQL. Units: Megabytes/second
`WriteIOPS`	The average number of disk write I/O operations per second. Units: Count/Second
`WriteLatency`	The average amount of time taken per disk I/O operation. Units: Seconds
`WriteThroughput`	The average number of bytes written to disk per second. Units: Bytes/Second

Amazon Aurora Metrics

The `AWS/RDS` namespace includes the following metrics that apply to database entities running on Amazon Aurora.

Metric	Description	Applies to
`ActiveTransactions`	The average number of current transactions executing on an Aurora database instance per second.	Aurora MySQL
`AuroraBinlogReplicaLag`	The amount of time a replica DB cluster running on Aurora with MySQL compatibility lags behind the source DB cluster. This metric reports the value of the `Seconds_Behind_Master` field of the MySQL `SHOW SLAVE STATUS` command. This metric is useful for monitoring replica lag between Aurora DB clusters that are replicating across different AWS Regions. For more information, see Aurora MySQL Replication.	Aurora MySQL
`AuroraReplicaLag`	For an Aurora Replica, the amount of lag when replicating updates from the primary instance, in milliseconds.	Aurora MySQL and Aurora PostgreSQL
`AuroraReplicaLagMaximum`	The maximum amount of lag between the primary instance and each Aurora DB instance in the DB cluster, in milliseconds.	Aurora MySQL and Aurora PostgreSQL
`AuroraReplicaLagMinimum`	The minimum amount of lag between the primary instance and each Aurora DB instance in the DB cluster, in milliseconds.	Aurora MySQL and Aurora PostgreSQL
`BinLogDiskUsage`	The amount of disk space occupied by binary logs on the master, in bytes.	Aurora MySQL

Metric	Description	Applies to
`BacktrackChangeRecordsCreationRate`	The number of backtrack change records created over five minutes for your DB cluster.	Aurora MySQL
`BacktrackChangeRecordsStored`	The actual number of backtrack change records used by your DB cluster.	Aurora MySQL
`BacktrackWindowActual`	The difference between the target backtrack window and the actual backtrack window.	Aurora MySQL
`BacktrackWindowAlert`	The number of times that the actual backtrack window is smaller than the target backtrack window for a given period of time.	Aurora MySQL
`BlockedTransactions`	The average number of transactions in the database that are blocked per second.	Aurora MySQL
`BufferCacheHitRatio`	The percentage of requests that are served by the buffer cache.	Aurora MySQL and Aurora PostgreSQL
`CommitLatency`	The amount of latency for commit operations, in milliseconds.	Aurora MySQL and Aurora PostgreSQL
`CommitThroughput`	The average number of commit operations per second.	Aurora MySQL and Aurora PostgreSQL
`CPUCreditBalance`	The number of CPU credits that an instance has accumulated. This metric applies only to `db.t2.small` and `db.t2.medium` instances. It is used to determine how long an Aurora MySQL DB instance can burst beyond its baseline performance level at a given rate. **Note** CPU credit metrics are reported at 5-minute intervals.	Aurora MySQL
`CPUCreditUsage`	The number of CPU credits consumed during the specified period. This metric applies only to `db.t2.small` and `db.t2.medium` instances. It identifies the amount of time during which physical CPUs have been used for processing instructions by virtual CPUs allocated to the Aurora MySQL DB instance. **Note** CPU credit metrics are reported at 5-minute intervals.	Aurora MySQL
`CPUUtilization`	The percentage of CPU used by an Aurora DB instance.	Aurora MySQL and Aurora PostgreSQL
`DatabaseConnections`	The number of connections to an Aurora DB instance.	Aurora MySQL and Aurora PostgreSQL
`DDLLatency`	The amount of latency for data definition language (DDL) requests, in milliseconds—for example, create, alter, and drop requests.	Aurora MySQL

Metric	Description	Applies to
`DDLThroughput`	The average number of DDL requests per second.	Aurora MySQL
`Deadlocks`	The average number of deadlocks in the database per second.	Aurora MySQL and Aurora PostgreSQL
`DeleteLatency`	The amount of latency for delete queries, in milliseconds.	Aurora MySQL
`DeleteThroughput`	The average number of delete queries per second.	Aurora MySQL
`DiskQueueDepth`	The number of outstanding read/write requests waiting to access the disk.	Aurora PostgreSQL
`DMLLatency`	The amount of latency for inserts, updates, and deletes, in milliseconds.	Aurora MySQL
`DMLThroughput`	The average number of inserts, updates, and deletes per second.	Aurora MySQL
`EngineUptime`	The amount of time that the instance has been running, in seconds.	Aurora MySQL and Aurora PostgreSQL
`FreeableMemory`	The amount of available random access memory, in bytes.	Aurora MySQL and Aurora PostgreSQL
`FreeLocalStorage`	The amount of storage available for temporary tables and logs, in bytes. Unlike for other DB engines, for Aurora DB instances this metric reports the amount of storage available to each DB instance for temporary tables and logs. This value depends on the DB instance class (for pricing information, see the Amazon RDS product page). You can increase the amount of free storage space for an instance by choosing a larger DB instance class for your instance.	Aurora MySQL and Aurora PostgreSQL
`InsertLatency`	The amount of latency for insert queries, in milliseconds.	Aurora MySQL
`InsertThroughput`	The average number of insert queries per second.	Aurora MySQL
`LoginFailures`	The average number of failed login attempts per second.	Aurora MySQL
`MaximumUsedTransactionIDs`	The age of the oldest unvacuumed transaction ID, in transactions. If this value reaches 2,146,483,648 (2^31 - 1,000,000), the database is forced into read-only mode, to avoid transaction ID wraparound. For more information, see Preventing Transaction ID Wraparound Failures in the PostgreSQL documentation.	Aurora PostgreSQL

Metric	Description	Applies to
NetworkReceiveThroughput	The amount of network throughput received from clients by each instance in the Aurora MySQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the Aurora DB cluster and the cluster volume.	Aurora MySQL and Aurora PostgreSQL
NetworkThroughput	The amount of network throughput both received from and transmitted to clients by each instance in the Aurora MySQL DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the DB cluster and the cluster volume.	Aurora MySQL and Aurora PostgreSQL
NetworkTransmitThroughput	The amount of network throughput sent to clients by each instance in the Aurora DB cluster, in bytes per second. This throughput doesn't include network traffic between instances in the DB cluster and the cluster volume.	Aurora MySQL and Aurora PostgreSQL
Queries	The average number of queries executed per second.	Aurora MySQL
RDSToAuroraPostgreSQLReplicaLag	The amount of lag in seconds when replicating updates from the primary RDS PostgreSQL instance to other nodes in the cluster.	Aurora PostgreSQL
ReadIOPS	The average number of disk I/O operations per second. Aurora with PostgreSQL compatibility reports read and write IOPS separately, in 1-minute intervals.	Aurora PostgreSQL
ReadLatency	The average amount of time taken per disk I/O operation.	Aurora PostgreSQL
ReadThroughput	The average number of bytes read from disk per second.	Aurora PostgreSQL
ResultSetCacheHitRatio	The percentage of requests that are served by the Resultset cache.	Aurora MySQL
SelectLatency	The amount of latency for select queries, in milliseconds.	Aurora MySQL
SelectThroughput	The average number of select queries per second.	Aurora MySQL
SwapUsage	The amount of swap space used on the Aurora PostgreSQL DB instance.	Aurora PostgreSQL
TransactionLogsDiskUsage	The amount of disk space occupied by transaction logs on the Aurora PostgreSQL DB instance.	Aurora PostgreSQL
UpdateLatency	The amount of latency for update queries, in milliseconds.	Aurora MySQL
UpdateThroughput	The average number of update queries per second.	Aurora MySQL

Metric	Description	Applies to
`VolumeBytesUsed`	The amount of storage used by your Aurora DB instance, in bytes. This value affects the cost of the Aurora DB cluster (for pricing information, see the Amazon RDS product page).	Aurora MySQL and Aurora PostgreSQL
`VolumeReadIOPS`	The number of billed read I/O operations from a cluster volume, reported at 5-minute intervals. Billed read operations are calculated at the cluster volume level, aggregated from all instances in the Aurora DB cluster, and then reported at 5-minute intervals. The value is calculated by taking the value of the **Read operations** metric over a 5-minute period. You can determine the amount of billed read operations per second by taking the value of the **Billed read operations** metric and dividing by 300 seconds. For example, if the **Billed read operations** returns 13,686, then the billed read operations per second is 45 (13,686 / 300 = 45.62). You accrue billed read operations for queries that request database pages that aren't in the buffer cache and therefore must be loaded from storage. You might see spikes in billed read operations as query results are read from storage and then loaded into the buffer cache.	Aurora MySQL and Aurora PostgreSQL
`VolumeWriteIOPS`	The number of write disk I/O operations to the cluster volume, reported at 5-minute intervals. See the description of VolumeReadIOPS above for a detailed description of how billed write operations are calculated.	Aurora MySQL and Aurora PostgreSQL
`WriteIOPS`	The average number of disk I/O operations per second. Aurora PostgreSQL reports read and write IOPS separately, on 1-minute intervals.	Aurora PostgreSQL
`WriteLatency`	The average amount of time taken per disk I/O operation.	Aurora PostgreSQL
`WriteThroughput`	The average number of bytes written to disk per second.	Aurora PostgreSQL

Amazon RDS Performance Insights Metrics

The `AWS/RDS` namespace includes the following metrics that apply to database entities running Amazon RDS Performance Insights.

Metric	Description	
DBLoad	The average number of active sessions for the DB engine.	
DBLoadCPU	The number of active sessions where the wait event type is CPU.	
DBLoadNonCPU	The number of active sessions where the wait event type is not CPU.	

Dimensions for RDS Metrics

Amazon RDS data can be filtered along any of the following dimensions in the table below.

Dimension	Description
DBInstanceIdentifier	This dimension filters the data you request for a specific database instance.
DBClusterIdentifier	This dimension filters the data you request for a specific Amazon Aurora DB cluster.
DBClusterIdentifier, Role	This dimension filters the data you request for a specific Amazon Aurora DB cluster, aggregating the metric by instance role (WRITER/ READER). For example, you can aggregate metrics for all READER instances that belong to a cluster.
DatabaseClass	This dimension filters the data you request for all instances in a database class. For example, you can aggregate metrics for all instances that belong to the database class db.m1.small
EngineName	This dimension filters the data you request for the identified engine name only. For example, you can aggregate metrics for all instances that have the engine name mysql.

Route 53 Metrics and Dimensions

Route 53 sends metrics to CloudWatch. CloudWatch provides detailed monitoring of Route 53 by default. Route 53 sends one-minute metrics to CloudWatch. For more information, see Monitoring Health Checks Using Amazon CloudWatch in the *Amazon Route 53 Developer Guide*.

> **Note**
> To get Route 53 metrics using CloudWatch, you must choose US East (N. Virginia) as the region. Route 53 metrics are not available if you select any other region. You can also optionally specify a Region dimension. For more information, see Dimensions for Route 53 Metrics (p. 287).

Route 53 Metrics

The AWS/Route53 namespace includes the following metrics.

Metric	Description
ChildHealthCheckHealthyCount	For a calculated health check, the number of health checks that are healthy among the health checks that Route 53 is monitoring. Valid statistics: Average (recommended), Minimum, Maximum Units: Healthy health checks
ConnectionTime	The average time, in milliseconds, that it took Route 53 health checkers to establish a TCP connection with the endpoint. You can view ConnectionTime for a health check either across all regions or for a selected geographic region. Valid statistics: Average (recommended), Minimum, Maximum Units: Milliseconds
HealthCheckPercentageHealthy	The percentage of Route 53 health checkers that consider the selected endpoint to be healthy. You can view HealthCheckPercentageHealthy only across all regions; data is not available for a selected region. Valid statistics: Average, Minimum, Maximum Units: Percent
HealthCheckStatus	The status of the health check endpoint that CloudWatch is checking. **1** indicates healthy, and **0** indicates unhealthy. You can view HealthCheckStatus only across all regions; data is not available for a selected region. Valid statistics: Minimum Units: none
SSLHandshakeTime	The average time, in milliseconds, that it took Route 53 health checkers to complete the SSL handshake. You can view SSLHandshakeTime for a health check either across all regions or for a selected geographic region. Valid statistics: Average (recommended), Minimum, Maximum Units: Milliseconds
TimeToFirstByte	The average time, in milliseconds, that it took Route 53 health checkers to receive the first byte of the response to an HTTP or HTTPS request. You can view TimeToFirstByte for a health check either across all regions or for a selected geographic region. Valid statistics: Average (recommended), Minimum, Maximum Units: Milliseconds

Dimensions for Route 53 Metrics

Route 53 metrics use the `AWS/Route53` namespace and provide metrics for `HealthCheckId`. When retrieving metrics, you must supply the `HealthCheckId` dimension.

In addition, for `ConnectionTime`, `SSLHandshakeTime`, and `TimeToFirstByte`, you can optionally specify `Region`. If you omit `Region`, CloudWatch returns metrics across all regions. If you include `Region`, CloudWatch returns metrics only for the specified region.

For more information, see Monitoring Health Checks Using CloudWatch in the *Amazon Route 53 Developer Guide*.

Amazon SageMaker Metrics and Dimensions

Amazon SageMaker sends metrics to CloudWatch. For more information, see Monitoring Amazon SageMaker with Amazon CloudWatch in the *Amazon SageMaker Developer Guide*.

Invocation Metrics

The `AWS/SageMaker` namespace includes the following request metrics.

Metric	Description
`Invocation4XXErrors`	The number of `InvokeEndpoint` requests where the model returned a 4xx HTTP response code. For each 4xx response, 1 is sent; otherwise, 0 is sent. Units: Count Valid statistics: Average, Sum
`Invocation5XXErrors`	The number of `InvokeEndpoint` requests where the model returned a 5xx HTTP response code. For each 5xx response, 1 is sent; otherwise, 0 is sent. Units: Count Valid statistics: Average, Sum
`Invocations`	The `number of InvokeEndpoint` requests sent to a model. To get the total number of requests to the endpoint variant, use the Sum statistic. Units: Count Valid statistics: Sum, Sample Count
`InvocationsPerInstance`	The number of invocations sent to a model, normalized by `InstanceCount` in each ProductionVariant. 1/`numberOfInstances` is sent as the value on each request, where `numberOfInstances` is the number of active instances for the ProductionVariant behind the endpoint at the time of the request. Units: Count Valid statistics: Sum
`ModelLatency`	The interval of time taken by a model to respond as viewed from . This interval includes the local communication times taken to send the request

Metric	Description
	and to fetch the response from container of a model and the time taken to complete the inference in the container. Units: Microseconds Valid statistics: Average, Sum, Min, Max, Sample Count
OverheadLatency	The interval of time added to the time taken to respond to a client request by overheads. This interval is measured from the time receives the request until it returns a response to the client, minus the ModelLatency. Overhead latency can vary depending on multiple factors, including request and response payload sizes, request frequency, and authentication/authorization of the request. Units: Microseconds Valid statistics: Average, Sum, Min, Max, Sample Count

Dimensions for Invocation Metrics

Dimension	Description
EndpointName, VariantName	Filters invocation metrics for a ProductionVariant of the specified endpoint and variant.

Instance Metrics

The `/aws/sagemaker/Endpoints` and `/aws/sagemaker/TrainingJobs` namespaces include the following metrics.

Metric	Description
CPUUtilization	The percentage of CPU units that are used by the containers on an instance. The value can range between 0 and 100, and is multiplied by the number of CPUs. For example, if there are four CPUs, CPUUtilization can range from 0% to 400%. For endpoint variants, the value is the sum of the CPU utilization of the primary and supplementary containers on the instance. For training jobs, the value is the CPU utilization of the Algorithm container on the instance. Units: Percent Valid statistics: Max
MemoryUtilizaton	The percentage of memory that is used by the containers on an instance. This value can range between 0% and 100%. For endpoint variants, the value is the sum of the memory utilization of the primary and supplementary containers on the instance. For training jobs, the value is the memory utilization of the Algorithm container on the instance.

Metric	Description
	Units: Percent
	Valid statistics: Max
`GPUUtilization`	The percentage of GPU units that are used by the containers on an instance. The value can range between 0 and 100 and is multiplied by the number of GPUs. For example, if there are four GPUs, `GPUUtilization` can range from 0% to 400%.
	For endpoint variants, the value is the sum of the GPU utilization of the primary and supplementary containers on the instance.
	For training jobs, the value is the GPU utilization of the Algorithm container on the instance.
	Units: Percent
	Valid statistics: Max

Dimensions for Host Metrics

Dimension	Description
`Host`	For endpoints, the value for this dimension has the format `[endpoint-name]/[ production-variant-name ]/[instance-id]`. Use this dimension to filter instance metrics for the specified endpoint, variant, and instance. This dimension format is present only in the `/aws/sagemaker/ Endpoints` namespace.
	For training jobs, the value for this dimension has the format `[training-job-name]/algo-[instance-number-in-cluster]`. Use this dimension to filter instance metrics for the specified training job and instance. This dimension format is present only in the `/aws/sagemaker/ TrainingJobs` namespace.

AWS Service Catalog Metrics and Dimensions

AWS Service Catalog sends the following metrics to CloudWatch.

AWS Service Catalog Metrics

The `AWS/ServiceCatalog` namespace includes the following metrics.

Metric	Description
`ProvisionedProductLaunch`	The number of provisioned products launched for a given product and provisioning artifact in a specified time period.
	Units: Count
	Valid statistics: Minimum, Maximum, Sum, Average

Dimensions for AWS Service Catalog Metrics

AWS Service Catalog sends the following dimensions to CloudWatch.

Dimension	Description
`State`	This dimension filters the data you request for all provisioned products launched with this specified state. This helps you categorize your data by the state of launch. Valid State: SUCCEEDED, FAILED
`ProductId`	This dimension filters the data you request for the identified product id only. This helps you to pinpoint an exact product from which to be launched.
`ProvisioningArtifactId`	This dimension filters the data you request for the identified provisioning artifact id only. This helps you to pinpoint an exact version of products from which to be launched.

Amazon Simple Email Service Metrics and Dimensions

Amazon Simple Email Service sends certain data points to CloudWatch. These data points track important metrics related to your email sending activities. For more information, see Retrieving Amazon SES Event Data from CloudWatch in the *Amazon Simple Email Service Developer Guide*.

Amazon SES Metrics

The following metrics are available from Amazon SES.

Metric	Description
Send	The number of messages that Amazon SES accepted and attempted to send. This value may be distinct from the Delivery metric, because messages could bounce or be rejected by the email servers that receive them. Unit: Count
Reject	The number of messages that Amazon SES did not send because they contained viruses or malicious content. Amazon SES may initially accept a message that contains a virus, but as soon as it detects the virus, it stops processing the message. When Amazon SES rejects a message, it doesn't attempt to deliver the message to the recipient's mail server. Unit: Count

Metric	Description
Bounce	The number of emails that resulted in a hard bounce. A hard bounce occurs when an email is permanently rejected by the recipient's mail server. Unit: Count
Complaint	The number of emails that were marked by their recipients as spam. Unit: Count
Delivery	The number of emails that Amazon SES successfully delivered to the mail servers of the intended recipients. Unit: Count
Open	The number of emails that were opened by their recipients. Amazon SES only tracks this metric when you use a configuration set that publishes Open events. Additionally, Amazon SES only captures Open events for HTML emails. Unit: Count
Click	The number of emails in which the recipient clicked a link. Amazon SES only tracks this metric when you use a configuration set that publishes Click events. Additionally, Amazon SES only captures Click events for HTML emails. Unit: Count
Rendering Failure	The number of emails that weren't able to be sent because they contained template rendering issues. This event type only occurs when you send templated email using the `SendTemplatedEmail` or `SendBulkTemplatedEmail` API operations. This event type can occur when template data is missing, or when there is a mismatch between template parameters and data. Unit: Count
Reputation.BounceRate	The percentage of messages that resulted in hard bounces. To calculate the percentage of emails that bounced, multiply `Reputation.BounceRate` by 100. Unit: Percent
Reputation.ComplaintRate	The percentage of messages that were reported as spam by their recipients. To calculate the percentage of emails that resulted in complaints, multiply `Reputation.ComplaintRate` by 100. Unit: Percent

Dimensions for Amazon SES Metrics

CloudWatch uses the dimension names that you specify when you add a CloudWatch event destination to a configuration set in Amazon SES. For more information, see Set Up a CloudWatch Event Destination for Amazon SES Event Publishing.

Amazon Simple Notification Service Metrics and Dimensions

Amazon Simple Notification Service sends data points to CloudWatch for several metrics. All active topics automatically send five-minute metrics to CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon Simple Notification Service. A topic stays active for six hours from the last activity (for example, any API call) on the topic. For more information, see Monitoring Amazon SNS with Amazon CloudWatch in the *Amazon Simple Notification Service Developer Guide*.

Amazon SNS Metrics

Amazon SNS sends the following metrics to CloudWatch.

Metric	Description
NumberOfMessagesPublished	The number of messages published to your Amazon SNS topics. Units: *Count* Valid Statistics: Sum
NumberOfNotificationsDelivered	The number of messages successfully delivered from your Amazon SNS topics to subscribing endpoints. For a delivery attempt to succeed, the endpoint's subscription must accept the message. A subscription accepts a message if a.) it lacks a filter policy or b.) its filter policy includes attributes that match those assigned to the message. If the subscription rejects the message, the delivery attempt is not counted for this metric. Units: *Count* Valid Statistics: Sum
NumberOfNotificationsFailed	The number of messages that Amazon SNS failed to deliver. For Amazon SQS, email, SMS, or mobile push endpoints, the metric increments by 1 when Amazon SNS stops attempting message deliveries. For HTTP or HTTPS endpoints, the metric includes every failed delivery attempt, including retries that follow the initial attempt. For all other endpoints, the count increases by 1 when the message fails to deliver (regardless of the number of attempts).

Metric	Description
	This metric does not include messages that were rejected by subscription filter policies. You can control the number of retries for HTTP endpoints. For more information, see Setting Amazon SNS Delivery Retry Policies for HTTP/HTTPS Endpoints. Units: *Count* Valid Statistics: Sum, Average
`NumberOfNotificationsFilteredOut`	The number of messages that were rejected by subscription filter policies. A filter policy rejects a message when the message attributes do not match the policy attributes. Units: *Count* Valid Statistics: Sum, Average
`NumberOfNotificationsFilteredOut-` `NoMessageAttributes`	The number of messages that were rejected by subscription filter policies because the messages have no attributes. Units: *Count* Valid Statistics: Sum, Average
`NumberOfNotificationsFilteredOut-` `InvalidAttributes`	The number of messages that were rejected by subscription filter policies because the message's attributes are invalid – for example, because the attribute JSON is incorrectly formatted. Units: *Count* Valid Statistics: Sum, Average
`PublishSize`	The size of messages published. Units: *Bytes* Valid Statistics: Minimum, Maximum, Average and Count

Metric	Description
SMSMonthToDateSpentUSD	The charges you have accrued since the start of the current calendar month for sending SMS messages.

You can set an alarm for this metric to know when your month-to-date charges are close to the monthly SMS spend limit for your account. When Amazon SNS determines that sending an SMS message would incur a cost that exceeds this limit, it stops publishing SMS messages within minutes.

For information about setting your monthly SMS spend limit, or for information about requesting a spend limit increase with AWS, see Setting SMS Messaging Preferences.

Units: *USD*

Valid Statistics: Maximum |
| SMSSuccessRate | The rate of successful SMS message deliveries.

Units: *Count*

Valid Statistics: Sum, Average, Data Samples |

Dimensions for Amazon Simple Notification Service Metrics

Amazon Simple Notification Service sends the following dimensions to CloudWatch.

Dimension	Description
Application	Filters on application objects, which represent an app and device registered with one of the supported push notification services, such as and .
Application,Platform	Filters on application and platform objects, where the platform objects are for the supported push notification services, such as and .
Country	Filters on the destination country or region of an SMS message. The country or region is represented by its ISO 3166-1 alpha-2 code.
Platform	Filters on platform objects for the push notification services, such as and .
TopicName	Filters on Amazon SNS topic names.
SMSType	Filters on the message type of SMS message. Can be *promotional* or *transactional*.

Amazon SQS Metrics and Dimensions

Amazon SQS sends data points to CloudWatch for several metrics. All active queues automatically send five-minute metrics to CloudWatch. A queue stays active for six hours from the last activity (for example, any API call) on the queue. For more information, see Monitoring Amazon SQS with Amazon CloudWatch in the *Amazon Simple Queue Service Developer Guide*.

> **Note**
> Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon SQS. Making requests to CloudWatch at this resolution might return no data.

Amazon SQS Metrics

The `AWS/SQS` namespace includes the following metrics.

Metric	Description
`ApproximateAgeOfOldestMessage`	The approximate age of the oldest non-deleted message in the queue. **Note** For dead-letter queues, the value of `ApproximateAgeOfOldestMessage` is the longest time that a message has been in the queue. Units: *Seconds* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)
`ApproximateNumberOfMessagesDelayed`	The number of messages in the queue that are delayed and not available for reading immediately. This can happen when the queue is configured as a delay queue or when a message has been sent with a delay parameter. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)
`ApproximateNumberOfMessagesNotVisible`	The number of messages that are in flight. Messages are considered to be *in flight* if they have been sent to a client but have not yet been deleted or have not yet reached the end of their visibility window. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)

Metric	Description
ApproximateNumberOfMessagesVisible	The number of messages available for retrieval from the queue. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)
NumberOfEmptyReceives	The number of `ReceiveMessage` API calls that did not return a message. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)
NumberOfMessagesDeleted	The number of messages deleted from the queue. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console) Amazon SQS emits the `NumberOfMessagesDeleted` metric for every successful deletion operation that uses a valid receipt handle, including duplicate deletions. The following scenarios might cause the value of the `NumberOfMessagesDeleted` metric to be higher than expected: • Calling the `DeleteMessage` action on different receipt handles that belong to the same message: If the message is not processed before the visibility timeout expires, the message becomes available to other consumers that can process it and delete it again, increasing the value of the `NumberOfMessagesDeleted` metric. • Calling the `DeleteMessage` action on the same receipt handle: If the message is processed and deleted but you call the `DeleteMessage` action again using the same receipt handle, a success status is returned, increasing the value of the `NumberOfMessagesDeleted` metric.

Metric	Description
`NumberOfMessagesReceived`	The number of messages returned by calls to the `ReceiveMessage` action. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)
`NumberOfMessagesSent`	The number of messages added to a queue. Units: *Count* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console)
`SentMessageSize`	The size of messages added to a queue. Units: *Bytes* Valid Statistics: Average, Minimum, Maximum, Sum, Data Samples (displays as Sample Count in the Amazon SQS console) Note that `SentMessageSize` does not display as an available metric in the CloudWatch console until at least one message is sent to the corresponding queue.

Dimensions for Amazon SQS Metrics

The only dimension that Amazon SQS sends to CloudWatch is `QueueName`. This means that all available statistics are filtered by `QueueName`.

Amazon Simple Storage Service Metrics and Dimensions

Amazon Simple Storage Service sends data points to CloudWatch for several metrics, such as object counts and bytes stored, once a day. For more information, see Monitoring Amazon S3 with CloudWatch in the *Amazon Simple Storage Service Developer Guide*.

Amazon S3 CloudWatch Daily Storage Metrics for Buckets

The `AWS/S3` namespace includes the following daily storage metrics for buckets.

Metric	Description
BucketSizeBytes	The amount of data in bytes stored in a bucket in the Standard storage class, Standard - Infrequent Access (Standard_IA) storage class, OneZone - Infraquent Access (OneZone_IA), Reduced Redundancy Storage (RRS) class, or Glacier (GLACIER) storage class Valid storage type filters: `StandardStorage`, `GlacierS3ObjectOverhead`, `StandardIAStorage`, `StandardIAObjectOverhead`, `OneZoneIAStorage`, `OneZoneIAObjectOverhead`, `ReducedRedundancyStorage`, `GlacierStorage`, and `GlacierObjectOverhead` (see the `StorageType` dimension) Units: Bytes Valid statistics: Average
NumberOfObjects	The total number of objects stored in a bucket for all storage classes Valid storage type filters: `AllStorageTypes` (see the `StorageType` dimension) Units: Count Valid statistics: Average

The `AWS/S3` namespace includes the following request metrics.

Amazon S3 CloudWatch Request Metrics

Metric	Description
AllRequests	The total number of HTTP requests made to an Amazon S3 bucket, regardless of type. If you're using a metrics configuration with a filter, then this metric only returns the HTTP requests made to the objects in the bucket that meet the filter's requirements. Units: Count Valid statistics: Sum
GetRequests	The number of HTTP GET requests made for objects in an Amazon S3 bucket. This doesn't include list operations. Units: Count Valid statistics: Sum **Note** Paginated list-oriented requests, like List Multipart Uploads, List Parts, Get Bucket Object versions, and others, are not included in this metric.
PutRequests	The number of HTTP PUT requests made for objects in an Amazon S3 bucket.

Metric	Description
	Units: Count
	Valid statistics: Sum
DeleteRequests	The number of HTTP DELETE requests made for objects in an Amazon S3 bucket. This also includes Delete Multiple Objects requests. This metric shows the number of requests, not the number of objects deleted.
	Units: Count
	Valid statistics: Sum
HeadRequests	The number of HTTP HEAD requests made to an Amazon S3 bucket.
	Units: Count
	Valid statistics: Sum
PostRequests	The number of HTTP POST requests made to an Amazon S3 bucket.
	Units: Count
	Valid statistics: Sum
	Note Delete Multiple Objects and SELECT Object Content requests are not included in this metric.
SelectRequests	The number of Amazon S3 SELECT Object Content requests made for objects in an Amazon S3 bucket.
	Units: Count
	Valid statistics: Sum
SelectScannedBytes	The number of bytes of data scanned with Amazon S3 SELECT Object Content requests in an Amazon S3 bucket.
	Units: Bytes
	Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max
SelectReturnedBytes	The number of bytes of data returned with Amazon S3 SELECT Object Content requests in an Amazon S3 bucket.
	Units: Bytes
	Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max
ListRequests	The number of HTTP requests that list the contents of a bucket.
	Units: Count
	Valid statistics: Sum

Metric	Description
BytesDownloaded	The number bytes downloaded for requests made to an Amazon S3 bucket, where the response includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max
BytesUploaded	The number bytes uploaded that contain a request body, made to an Amazon S3 bucket. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max
4xxErrors	The number of HTTP 4xx client error status code requests made to an Amazon S3 bucket with a value of either 0 or 1. The average statistic shows the error rate, and the sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count
5xxErrors	The number of HTTP 5xx server error status code requests made to an Amazon S3 bucket with a value of either 0 or 1. The average statistic shows the error rate, and the sum statistic shows the count of that type of error, during each period. Units: Counts Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count
FirstByteLatency	The per-request time from the complete request being received by an Amazon S3 bucket to when the response starts to be returned. Units: Milliseconds Valid statistics: Average, Sum, Min, Max, Sample Count
TotalRequestLatency	The elapsed per-request time from the first byte received to the last byte sent to an Amazon S3 bucket. This includes the time taken to receive the request body and send the response body, which is not included in FirstByteLatency. Units: Milliseconds Valid statistics: Average, Sum, Min, Max, Sample Count

Amazon S3 CloudWatch Dimensions

The following dimensions are used to filter Amazon S3 metrics.

Dimension	Description
BucketName	This dimension filters the data you request for the identified bucket only.
StorageType	This dimension filters the data that you have stored in a bucket by the type of storage. The types are StandardStorage for the STANDARD storage class, StandardIAStorage for the STANDARD_IA storage class, OneZoneIAStorage for the ONEZONE_IA storage class, ReducedRedundancyStorage for the REDUCED_REDUNDANCY storage class, GlacierStorage for the GLACIER storage class, and AllStorageTypes. The AllStorageTypes type includes the STANDARD, STANDARD_IA, ONEZONE_IA, REDUCED_REDUNDANCY, and GLACIER storage classes.
FilterId	This dimension filters metrics configurations that you specify for request metrics on a bucket, for example, a prefix or a tag. You specify a filter id when you create a metrics configuration. For more information, see Metrics Configurations for Buckets.

AWS Shield Advanced Metrics

For information about the Shield Advanced metrics that you can use with CloudWatch, see Monitoring with Amazon CloudWatch. in the AWS WAF Developer Guide.

AWS Step Functions Metrics and Dimensions

The following metrics are available for AWS Step Functions. For more information, see Monitoring Step Functions Using CloudWatch in the *AWS Step Functions Developer Guide*.

Execution Metrics

The AWS/States namespace includes the following metrics for Step Functions executions:

Metric	Description
ExecutionTime	The interval, in milliseconds, between the time the execution starts and the time it closes.
ExecutionThrottled	The number of StateEntered events and retries that have been throttled. This is related to StateTransition throttling. For more information, see Limits Related to State Throttling in the *AWS Step Functions Developer Guide*.
ExecutionsAborted	The number of aborted or terminated executions.
ExecutionsFailed	The number of failed executions.
ExecutionsStarted	The number of started executions.
ExecutionsSucceeded	The number of successfully completed executions.
ExecutionsTimedOut	The number of executions that time out for any reason.

Dimension for Step Functions Execution Metrics

Dimension	Description
StateMachineArn	The ARN of the state machine for the execution in question.

Activity Metrics

The `AWS/States` namespace includes the following metrics for Step Functions activities:

Metric	Description
ActivityRunTime	The interval, in milliseconds, between the time the activity starts and the time it closes.
ActivityScheduleTime	The interval, in milliseconds, for which the activity stays in the schedule state.
ActivityTime	The interval, in milliseconds, between the time the activity is scheduled and the time it closes.
ActivitiesFailed	The number of failed activities.
ActivitiesHeartbeatTimedOut	The number of activities that time out due to a heartbeat timeout.
ActivitiesScheduled	The number of scheduled activities.
ActivitiesStarted	The number of started activities.
ActivitiesSucceeded	The number of successfully completed activities.
ActivitiesTimedOut	The number of activities that time out on close.

Dimension for Step Functions Activity Metrics

Dimension	Description
ActivityArn	The ARN of the activity.

Lambda Function Metrics

The `AWS/States` namespace includes the following metrics for Step Functions Lambda functions:

Metric	Description
LambdaFunctionRunTime	The interval, in milliseconds, between the time the Lambda function starts and the time it closes.
LambdaFunctionScheduleTime	The interval, in milliseconds, for which the Lambda function stays in the schedule state.

Metric	Description
LambdaFunctionTime	The interval, in milliseconds, between the time the Lambda function is scheduled and the time it closes.
LambdaFunctionsFailed	The number of failed Lambda functions.
LambdaFunctionsHeartbeatTimedOut	The number of Lambda functions that time out due to a heartbeat timeout.
LambdaFunctionsScheduled	The number of scheduled Lambda functions.
LambdaFunctionsStarted	The number of started Lambda functions.
LambdaFunctionsSucceeded	The number of successfully completed Lambda functions.
LambdaFunctionsTimedOut	The number of Lambda functions that time out on close.

Dimension for Step Functions Lambda Function Metrics

Dimension	Description
LambdaFunctionArn	The ARN of the Lambda function.

Amazon SWF Metrics and Dimensions

The `AWS/SWF` namespace includes the metrics in the following table. For more information, see Amazon SWF Metrics for CloudWatch in the *AWS Storage Gateway User Guide*.

Metric	Description
DecisionTaskScheduleToStartTime	The time interval, in milliseconds, between the time that the decision task was scheduled and when it was picked up by a worker and started. CloudWatch Units: `Time` Dimensions: `Domain, WorkflowTypeName, WorkflowTypeVersion` Valid statistics: `Average, Minimum, Maximum`
DecisionTaskStartToCloseTime	The time interval, in milliseconds, between the time that the decision task was started and when it closed. CloudWatch Units: `Time` Dimensions: `Domain, WorkflowTypeName, WorkflowTypeVersion` Valid statistics: `Average, Minimum, Maximum`
DecisionTasksCompleted	The count of decision tasks that have been completed. CloudWatch Units: `Count`

Metric	Description
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Sum`
`StartedDecisionTasksTimedOutOnClose`	The count of decision tasks that started but timed out on closing.
	CloudWatch Units: `Count`
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Sum`
`WorkflowStartToCloseTime`	The time, in milliseconds, between the time the workflow started and when it closed.
	CloudWatch Units: `Time`
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Average`, `Minimum`, `Maximum`
`WorkflowsCanceled`	The count of workflows that were canceled.
	CloudWatch Units: `Count`
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Sum`
`WorkflowsCompleted`	The count of workflows that completed.
	CloudWatch Units: `Count`
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Sum`
`WorkflowsContinuedAsNew`	The count of workflows that continued as new.
	CloudWatch Units: `Count`
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Sum`
`WorkflowsFailed`	The count of workflows that failed.
	CloudWatch Units: `Count`
	Dimensions: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`
	Valid statistics: `Sum`

Metric	Description
`WorkflowsTerminated`	The count of workflows that were terminated. CloudWatch Units: `Count` Dimensions: `Domain, WorkflowTypeName, WorkflowTypeVersion` Valid statistics: `Sum`
`WorkflowsTimedOut`	The count of workflows that timed out, for any reason. CloudWatch Units: `Count` Dimensions: `Domain, WorkflowTypeName, WorkflowTypeVersion` Valid statistics: `Sum`
`ActivityTaskScheduleToCloseTime`	The time interval, in milliseconds, between the time when the activity was scheduled and when it closed. CloudWatch Units: `Time` Dimensions: `Domain, ActivityTypeName, ActivityTypeVersion` Valid statistics: `Average, Minimum, Maximum`
`ActivityTaskScheduleToStartTime`	The time interval, in milliseconds, between the time when the activity task was scheduled and when it started. CloudWatch Units: `Time` Dimensions: `Domain, ActivityTypeName, ActivityTypeVersion` Valid statistics: `Average, Minimum, Maximum`
`ActivityTaskStartToCloseTime`	The time interval, in milliseconds, between the time when the activity task started and when it closed. CloudWatch Units: `Time` Dimensions: `Domain, ActivityTypeName, ActivityTypeVersion` Valid statistics: `Average, Minimum, Maximum`
`ActivityTasksCanceled`	The count of activity tasks that were canceled. CloudWatch Units: `Count` Dimensions: `Domain, ActivityTypeName, ActivityTypeVersion` Valid statistics: `Sum`

Metric	Description
`ActivityTasksCompleted`	The count of activity tasks that completed. CloudWatch Units: `Count` Dimensions: `Domain`, `ActivityTypeName`, `ActivityTypeVersion` Valid statistics: `Sum`
`ActivityTasksFailed`	The count of activity tasks that failed. CloudWatch Units: `Count` Dimensions: `Domain`, `ActivityTypeName`, `ActivityTypeVersion` Valid statistics: `Sum`
`ScheduledActivityTasksTimedOutOnClose`	The count of activity tasks that were scheduled but timed out on close. CloudWatch Units: `Count` Dimensions: `Domain`, `ActivityTypeName`, `ActivityTypeVersion` Valid statistics: `Sum`
`ScheduledActivityTasksTimedOutOnStart`	The count of activity tasks that were scheduled but timed out on start. CloudWatch Units: `Count` Dimensions: `Domain`, `ActivityTypeName`, `ActivityTypeVersion` Valid statistics: `Sum`
`StartedActivityTasksTimedOutOnClose`	The count of activity tasks that were started but timed out on close. CloudWatch Units: `Count` Dimensions: `Domain`, `ActivityTypeName`, `ActivityTypeVersion` Valid statistics: `Sum`
`StartedActivityTasksTimedOutOnHeartbeat`	The count of activity tasks that were started but timed out due to a heartbeat timeout. CloudWatch Units: `Count` Dimensions: `Domain`, `ActivityTypeName`, `ActivityTypeVersion` Valid statistics: `Sum`

Metric	Description
`ThrottledEvents`	The count of requests that have been throttled. CloudWatch Units: `Count` Dimensions: `APIName, DecisionName` Valid statistics: `Sum`
`ProvisionedBucketSize`	The count of available requests per second. Dimensions: `APIName, DecisionName` Valid statistics: `Minimum`
`ConsumedCapacity`	The count of requests per second. CloudWatch Units: `Count` Dimensions: `APIName, DecisionName` Valid statistics: `Sum`
`ProvisionedRefillRate`	The count of requests per second that are allowed into the bucket. Dimensions: `APIName, DecisionName` Valid statistics: `Minimum`

The Amazon SWF metrics support the following dimensions.

Dimension	Description
`Domain`	Filters data to the Amazon SWF domain that the workflow or activity is running in.
`ActivityTypeName`	Filters data to the name of the activity type.
`ActivityTypeVersion`	Filters data to the version of the activity type.
`WorkflowTypeName`	Filters data to the name of the workflow type for this workflow execution.
`WorkflowTypeVersion`	Filters data to the version of the workflow type for this workflow execution.
`APIName`	Filters data to an API of the specified API name.
`DecisionName`	Filters data to the specified Decision name.
`TaskListName`	Filters data to the specified Task List name.

AWS Storage Gateway Metrics and Dimensions

AWS Storage Gateway sends data points to CloudWatch for several metrics. All active queues automatically send five-minute metrics to CloudWatch. Detailed monitoring, or one-minute metrics, is

currently unavailable for AWS Storage Gateway. For more information, see Monitoring Your AWS Storage Gateway in the *AWS Storage Gateway User Guide*.

AWS Storage Gateway Metrics

The `AWS/StorageGateway` namespace includes the following metrics.

You can use these metrics to get information about your gateways. The entries in the table are grouped functionally by measure. Specify the `GatewayId` or `GatewayName` dimension for each metric to view the data for a gateway. Note that these metrics are measured in 5-minute intervals.

Metric	Description	Applies To..
CacheHitPercent	Percent of application reads served from the cache. The sample is taken at the end of the reporting period. Units: Percent	File, Cached volumes and Tape.
CachePercentUsed	Percent use of the gateway's cache storage. The sample is taken at the end of the reporting period. Units: Percent	File, Cached volumes and Tape.
CachePercentDirty	Percent of the gateway's cache that has not been persisted to AWS. The sample is taken at the end of the reporting period. Units: Percent	File, Cached volumes and Tape.
CloudBytesDownloaded	The total number of compressed bytes that the gateway downloaded from AWS during the reporting period. Use this metric with the `Sum` statistic to measure throughput and with the `Samples` statistic to measure input/output operations per second (IOPS). Units: Bytes	File, Cached volumes, Stored volumes and Tape.
CloudDownloadLatency	The total number of milliseconds spent reading data from AWS during the reporting period. Use this metric with the `Average` statistic to measure latency. Units: Milliseconds	File, Cached volumes, Stored volumes and Tape.

Metric	Description	Applies To..
CloudBytesUploaded	The total number of compressed bytes that the uploaded to AWS during the reporting period. Use this metric with the Sum statistic to measure throughput and with the Samples statistic to measure IOPS. Units: Bytes	File, Cached volumes, Stored volumes and Tape.
UploadBufferFree	The total amount of unused space in the gateway's upload buffer. The sample is taken at the end of the reporting period. Units: Bytes	Cached volumes and Tape.
CacheFree	The total amount of unused space in the gateway's cache storage. The sample is taken at the end of the reporting period. Units: Bytes	File, Cached volumes, and Tape.
UploadBufferPercentUsed	Percent use of the gateway's upload buffer. The sample is taken at the end of the reporting period. Units: Percent	Cached volumes and Tape.
UploadBufferUsed	The total number of bytes being used in the gateway's upload buffer. The sample is taken at the end of the reporting period. Units: Bytes	Cached volumes and Tape.
CacheUsed	The total number of bytes being used in the gateway's cache storage. The sample is taken at the end of the reporting period. Units: Bytes	File, Cached volumes and Tape.
QueuedWrites	The number of bytes waiting to be written to AWS, sampled at the end of the reporting period for all volumes in the gateway. These bytes are kept in your gateway's working storage. Units: Bytes	File, Cached volumes, Stored volumes and Tape.

Metric	Description	Applies To..
ReadBytes	The total number of bytes read from your on-premises applications in the reporting period for all volumes in the gateway. Use this metric with the Sum statistic to measure throughput and with the Samples statistic to measure IOPS. Units: Bytes	File, Cached volumes, and Stored volumes.
ReadTime	The total number of milliseconds spent to do read operations from your on-premises applications in the reporting period for all volumes in the gateway. Use this metric with the Average statistic to measure latency. Units: Milliseconds	File, Cached volumes, and Stored volumes.
TotalCacheSize	The total size of the cache in bytes. The sample is taken at the end of the reporting period. Units: Bytes	File, Cached volumes, and Tape.
WriteBytes	The total number of bytes written to your on-premises applications in the reporting period for all volumes in the gateway. Use this metric with the Sum statistic to measure throughput and with the Samples statistic to measure IOPS. Units: Bytes	File, Cached volumes, and Stored volumes.
WriteTime	The total number of milliseconds spent to do write operations from your on-premises applications in the reporting period for all volumes in the gateway. Use this metric with the Average statistic to measure latency. Units: Milliseconds	File, Cached volumes, and Stored volumes.

Metric	Description	Applies To..
TimeSinceLastRecoveryPoint	The time since the last available recovery point. For more information, see Your Cached Gateway is Unreachable and You Want to Recover Your Data. Units: Seconds	Cached volumes and Stored volumes.
WorkingStorageFree	The total amount of unused space in the gateway's working storage. The sample is taken at the end of the reporting period. Units: Bytes	Stored volumes only.
WorkingStoragePercentUsed	Percent use of the gateway's upload buffer. The sample is taken at the end of the reporting period. Units: Percent	Stored volumes only.
WorkingStorageUsed	The total number of bytes being used in the gateway's upload buffer. The sample is taken at the end of the reporting period. Units: Bytes	Stored volumes only.

The following table describes the AWS Storage Gateway metrics that you can use to get information about your storage volumes. Specify the `VolumeId` dimension for each metric to view the data for a storage volume.

Metric	Description	Cached volumes	Stored volumes
CacheHitPercent	Percent of application read operations from the volume that are served from cache. The sample is taken at the end of the reporting period. When there are no application read operations from the volume, this metric reports 100 percent. Units: Percent	yes	no
CachePercentDirty	The volume's contribution to the overall percentage of the gateway's cache	yes	no

Metric	Description	Cached volumes	Stored volumes
	that has not been persisted to AWS. The sample is taken at the end of the reporting period. Use the `CachePercentDirty` metric of the gateway to view the overall percentage of the gateway's cache that has not been persisted to AWS. For more information, see Monitoring Your Gateway. Units: Percent		
`CachePercentUsed`	The volume's contribution to the overall percent use of the gateway's cache storage. The sample is taken at the end of the reporting period. Use the `CachePercentUsed` metric of the gateway to view overall percent use of the gateway's cache storage. For more information, see Monitoring Your Gateway. Units: Percent	yes	no
`ReadBytes`	The total number of bytes read from your on-premises applications in the reporting period. Use this metric with the `Sum` statistic to measure throughput and with the `Samples` statistic to measure IOPS. Units: Bytes	yes	yes

Metric	Description	Cached volumes	Stored volumes
ReadTime	The total number of milliseconds spent to do read operations from your on-premises applications in the reporting period. Use this metric with the `Average` statistic to measure latency. Units: Milliseconds	yes	yes
WriteBytes	The total number of bytes written to your on-premises applications in the reporting period. Use this metric with the `Sum` statistic to measure throughput and with the `Samples` statistic to measure IOPS. Units: Bytes	yes	yes
WriteTime	The total number of milliseconds spent to do write operations from your on-premises applications in the reporting period. Use this metric with the `Average` statistic to measure latency. Units: Milliseconds	yes	yes
QueuedWrites	The number of bytes waiting to be written to AWS, sampled at the end of the reporting period. Units: Bytes	yes	yes

The following table describes the Storage Gateway metrics that you can use to get information about your file shares.

Metric	Description
CacheHitPercent	Percent of application read operations from the file shares that are served from cache. The sample is taken at the end of the reporting period. When there are no application read operations from the file share, this metric reports 100 percent. Units: Percent
CachePercentDirty	The file share's contribution to the overall percentage of the gateway's cache that has not been persisted to AWS. The sample is taken at the end of the reporting period. Use the CachePercentDirty metric of the gateway to view the overall percentage of the gateway's cache that has not been persisted to AWS. For more information, see Monitoring Your Gateway. Units: Percent
CachePercentUsed	The file share's contribution to the overall percent use of the gateway's cache storage. The sample is taken at the end of the reporting period. Use the CachePercentUsed metric of the gateway to view overall percent use of the gateway's cache storage. For more information, see Monitoring Your Gateway. Units: Percent
ReadBytes	The total number of bytes read from your on-premises applications in the reporting period for a file share. Use this metric with the Sum statistic to measure throughput and with the Samples statistic to measure IOPS. Units: Bytes
ReadTime	The total number of milliseconds spent to do read operations from your on-premises applications in the reporting period. Use this metric with the Average statistic to measure latency. Units: Milliseconds
WriteBytes	The total number of bytes written to your on-premises applications in the reporting period.

Metric	Description
	Use this metric with the `Sum` statistic to measure throughput and with the `Samples` statistic to measure IOPS. Units: Bytes
`WriteTime`	The total number of milliseconds spent to do write operations from your on-premises applications in the reporting period. Use this metric with the `Average` statistic to measure latency. Units: Milliseconds

Dimensions for AWS Storage Gateway Metrics

The Amazon CloudWatch namespace for the AWS Storage Gateway service is `AWS/StorageGateway`. Data is available automatically in 5-minute periods at no charge.

Dimension	Description
`GatewayId, GatewayName`	These dimensions filter the data you request to gateway-specific metrics. You can identify a gateway to work by its `GatewayId` or its `GatewayName`. However, note that if the name of your gateway was changed for the time range that you are interested in viewing metrics, then you should use the `GatewayId`. Throughput and latency data of a gateway is based on all the volumes for the gateway. For information about working with gateway metrics, see Measuring Performance Between Your Gateway and AWS.
`VolumeId`	This dimension filters the data you request to volume-specific metrics. Identify a storage volume to work with by its `VolumeId`. For information about working with volume metrics, see Measuring Performance Between Your Application and Gateway.

Amazon Translate Metrics

The following table describes the Amazon Translate metrics.

CloudWatch Metrics for Amazon Translate

Metric	Description
`CharacterCount`	The number of billable characters in requests. Valid dimensions: Language pair, Operation Valid statistics: Average, Maximum, Minimum, Sum

Metric	Description
	Unit: Count
`ResponseTime`	The time that it took to respond to a request.
	Valid dimensions: Language pair, Operation
	Valid statistics: Data samples, Average
	Unit: For Data samples, count. For Average statistics, milliseconds.
`ServerErrorCount`	The number of server errors. The HTTP response code range for a server error is 500 to 599.
	Valid dimension: Operation
	Valid statistics: Average, Sum
	Unit: Count
`SuccessfulRequestCount`	The number of successful translation requests. The response code for a successful request is 200 to 299.
	Valid dimension: Operation
	Valid statistics: Average, Sum
	Unit: Count
`ThrottledCount`	The number of requests subject to throttling. Use `ThrottledCount` to determine if your application is sending requests to Amazon Translate faster than your account is configured to accept them. For more information, see Amazon Translate Limits in the *Amazon Web Services General Reference*.
	Valid dimension: Operation
	Valid statistics: Average, Sum
	Unit: Count
`UserErrorCount`	The number of user errors that occurred. The HTTP response code range for a user error is 400 to 499.
	Valid dimension: Operation
	Valid statistics: Average, Sum
	Unit: Count

CloudWatch Dimensions for Amazon Translate

Use the following dimensions to filter Amazon Translate metrics. Metrics are grouped by the source language and the target language.

Dimension	Description
LanguagePair	Restricts the metrics to only those that contain the specified languages.
Operation	Restricts the metrics to only those with the specified operation.

AWS Trusted Advisor Metrics and Dimensions

Trusted Advisor sends metrics to CloudWatch. For more information, see Creating Trusted Advisor Metrics in CloudWatch in the *AWS Support User Guide*.

The `AWS/TrustedAdvisor` namespace includes the following metrics.

Trusted Advisor Check-Level Metrics

Metric	Description
RedResources	The number of resources identifed by a Trusted Advisor check that are in a "red" (ERROR) state.
YellowResources	The number of resources identifed by a Trusted Advisor check that are in a "yellow" (WARN) state.

Trusted Advisor Category-Level Metrics

Metric	Description
GreenChecks	The number of Trusted Advisor checks in a "green" (OK) state.
RedChecks	The number of Trusted Advisor checks in a "red" (ERROR) state.
YellowChecks	The number of Trusted Advisor checks in a "yellow" (WARN) state.

Trusted Advisor Service-Level Metrics

Metric	Description
ServiceLimitUsage	The percentage of resource utilization against a service limit.

Dimensions for Check-Level Metrics

Dimension	Description
CheckName	The name of a Trusted Advisor check. For more information about checks, see Cost Optimization.

Dimensions for Category-Level Metrics

Dimension	Description
`Category`	The name of a Trusted Advisor check category. For more information about check categories, see Trusted Advisor Best Practices (Checks).

Dimensions for Service Limit Metrics

Dimension	Description
`Region`	The region for a given service limit.
`ServiceName`	The name of a service.
`ServiceLimit`	The name of service limit. For more information about checks, see Service Limits Check Questions.

Amazon VPC NAT Gateway Metrics and Dimensions

NAT gateway metric data is provided at 1-minute frequency. For more information, see Monitoring Your NAT Gateway with Amazon CloudWatch in the *Amazon VPC User Guide*.

NAT Gateway Metrics

The following metrics are available from the NAT gateway service.

Metric	Description
ActiveConnectionCount	The total number of concurrent active TCP connections through the NAT gateway. A value of zero indicates that there are no active connections through the NAT gateway. Units: Count Statistics: The most useful statistic is `Max`.
BytesInFromDestination	The number of bytes received by the NAT gateway from the destination. If the value for `BytesOutToSource` is less than the value for `BytesInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Units: Bytes Statistics: The most useful statistic is `Sum`.

Metric	Description
BytesInFromSource	The number of bytes received by the NAT gateway from clients in your VPC.
	If the value for `BytesOutToDestination` is less than the value for `BytesInFromSource`, there may be data loss during NAT gateway processing.
	Units: Bytes
	Statistics: The most useful statistic is `Sum`.
BytesOutToDestination	The number of bytes sent out through the NAT gateway to the destination.
	A value greater than zero indicates that there is traffic going to the internet from clients that are behind the NAT gateway. If the value for `BytesOutToDestination` is less than the value for `BytesInFromSource`, there may be data loss during NAT gateway processing.
	Unit: Bytes
	Statistics: The most useful statistic is `Sum`.
BytesOutToSource	The number of bytes sent through the NAT gateway to the clients in your VPC.
	A value greater than zero indicates that there is traffic coming from the internet to clients that are behind the NAT gateway. If the value for `BytesOutToSource` is less than the value for `BytesInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway.
	Units: Bytes
	Statistics: The most useful statistic is `Sum`.
ConnectionAttemptCount	The number of connection attempts made through the NAT gateway.
	If the value for `ConnectionEstablishedCount` is less than the value for `ConnectionAttemptCount`, this indicates that clients behind the NAT gateway attempted to establish new connections for which there was no response.
	Unit: Count
	Statistics: The most useful statistic is `Sum`.

Metric	Description
ConnectionEstablishedCount	The number of connections established through the NAT gateway.
	If the value for `ConnectionEstablishedCount` is less than the value for `ConnectionAttemptCount`, this indicates that clients behind the NAT gateway attempted to establish new connections for which there was no response.
	Unit: Count
	Statistics: The most useful statistic is `Sum`.
ErrorPortAllocation	The number of times the NAT gateway could not allocate a source port.
	A value greater than zero indicates that too many concurrent connections are open through the NAT gateway.
	Units: Count
	Statistics: The most useful statistic is `Sum`.
IdleTimeoutCount	The number of connections that transitioned from the active state to the idle state. An active connection transitions to idle if it was not closed gracefully and there was no activity for the last 350 seconds.
	A value greater than zero indicates that there are connections that have been moved to an idle state. If the value for `IdleTimeoutCount` increases, it may indicate that clients behind the NAT gateway are re-using stale connections.
	Unit: Count
	Statistics: The most useful statistic is `Sum`.
PacketsDropCount	The number of packets dropped by the NAT gateway.
	A value greater than zero may indicate an ongoing transient issue with the NAT gateway. If this value is high, see the AWS service health dashboard.
	Units: Count
	Statistics: The most useful statistic is `Sum`.

Metric	Description
PacketsInFromDestination	The number of packets received by the NAT gateway from the destination. If the value for `PacketsOutToSource` is less than the value for `PacketsInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is `Sum`.
PacketsInFromSource	The number of packets received by the NAT gateway from clients in your VPC. If the value for `PacketsOutToDestination` is less than the value for `PacketsInFromSource`, there may be data loss during NAT gateway processing. Unit: Count Statistics: The most useful statistic is `Sum`.
PacketsOutToDestination	The number of packets sent out through the NAT gateway to the destination. A value greater than zero indicates that there is traffic going to the internet from clients that are behind the NAT gateway. If the value for `PacketsOutToDestination` is less than the value for `PacketsInFromSource`, there may be data loss during NAT gateway processing. Unit: Count Statistics: The most useful statistic is `Sum`.
PacketsOutToSource	The number of packets sent through the NAT gateway to the clients in your VPC. A value greater than zero indicates that there is traffic coming from the internet to clients that are behind the NAT gateway. If the value for `PacketsOutToSource` is less than the value for `PacketsInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is `Sum`.

Dimensions for NAT Gateway Metrics

You can filter the NAT gateway data using the following dimensions.

Dimension	Description
NatGatewayId	This dimension filters data by the NAT gateway ID.

Amazon VPC VPN Metrics and Dimensions

Amazon VPN sends data to CloudWatch as it becomes available. For more information, see Monitoring with CloudWatch in the *Amazon VPC User Guide*.

VPN Metrics

The following metrics are available from Amazon VPC VPN.

Metric	Description
TunnelState	The state of the tunnel. 0 indicates DOWN and 1 indicates UP. Units: Boolean
TunnelDataIn	The bytes received through the VPN tunnel. Each metric data point represents the number of bytes received after the previous data point. Use the Sum statistic to show the total number of bytes received during the period. This metric counts the data after decryption. Units: Bytes
TunnelDataOut	The bytes sent through the VPN tunnel. Each metric data point represents the number of bytes sent after the previous data point. Use the Sum statistic to show the total number of bytes sent during the period. This metric counts the data before encryption. Units: Bytes

Dimensions for VPN Metrics

You can filter the Amazon VPC VPN data using the following dimensions.

Dimension	Description
VpnId	This dimension filters the data by the VPN connection.
TunnelIpAddress	This dimension filters the data by the IP address of the tunnel for the virtual private gateway.

AWS WAF Metrics and Dimensions

AWS WAF sends data to CloudWatch every minute. For more information, see Testing Web ACLs in the *AWS WAF Developer Guide*.

AWS WAF Metrics

The `WAF` namespace includes the following metrics.

Metric	Description
`AllowedRequests`	The number of allowed web requests. Reporting criteria: There is a nonzero value Valid statistics: Sum
`BlockedRequests`	The number of blocked web requests. Reporting criteria: There is a nonzero value Valid statistics: Sum
`CountedRequests`	The number of counted web requests. Reporting criteria: There is a nonzero value A counted web request is one that matches all of the conditions in a particular rule. Counted web requests are typically used for testing. Valid statistics: Sum
`PassedRequests`	The number of passed requests for a rule group. Reporting criteria: There is a nonzero value Passed requests are requests that did not match any rule contained in the rule group. Valid statistics: Sum

AWS WAF Dimensions

AWS WAF for CloudFront can use the following dimension combinations:

- Rule, WebACL
- RuleGroup, WebACL
- Rule, RuleGroup

AWS WAF for Application Load Balancer can use the following dimension combinations:

- Region, Rule, WebACL
- Region, RuleGroup, WebACL
- Region, Rule, RuleGroup

Dimension	Description
Rule	One of the following: • The metric name of the Rule. • ALL, which represents all rules within a WebACL or RuleGroup. • Default_Action (only when combined with the WebACL dimension), which represents the action assigned to any request that does not match any rule with either an allow or block action.
RuleGroup	The metric name of the RuleGroup.
WebACL	The metric name of the WebACL.
Region	The region of the application load balancer.

Amazon WorkSpaces Metrics and Dimensions

Amazon WorkSpaces sends data points to CloudWatch for several metrics every five minutes (five-minute metrics). Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon WorkSpaces. For more information, see Monitoring Amazon WorkSpaces in the *Amazon WorkSpaces Administration Guide*.

Amazon WorkSpaces Metrics

The `AWS/WorkSpaces` namespace includes the following metrics.

Metric	Description	Dimensions	Statistics Available	Units
Available[1]	The number of WorkSpaces that returned a healthy status.	DirectoryId WorkspaceId	Average, Sum, Maximum, Minimum, Data Samples	Count
Unhealthy[1]	The number of WorkSpaces that returned an unhealthy status.	DirectoryId WorkspaceId	Average, Sum, Maximum, Minimum, Data Samples	Count
ConnectionAttempt[2]	The number of connection attempts.	DirectoryId WorkspaceId	Average, Sum, Maximum, Minimum, Data Samples	Count
ConnectionSuccess[2]	The number of successful connections.	DirectoryId WorkspaceId	Average, Sum, Maximum, Minimum, Data Samples	Count
ConnectionFailure[2]	The number of failed connections.	DirectoryId WorkspaceId	Average, Sum, Maximum,	Count

Metric	Description	Dimensions	Statistics Available	Units
			Minimum, Data Samples	
SessionLaunchTime[2]	The amount of time it takes to initiate a WorkSpaces session.	DirectoryID WorkspaceID	Average, Sum, Maximum, Minimum, Data Samples	Second (time)
InSessionLatency[2]	The round trip time between the WorkSpaces client and the WorkSpace.	DirectoryID WorkspaceID	Average, Sum, Maximum, Minimum, Data Samples	Millisecond (time)
SessionDisconnect[2]	The number of connections that were closed, including user-initiated and failed connections.	DirectoryID WorkspaceID	Average, Sum, Maximum, Minimum, Data Samples	Count
UserConnected[3]	The number of WorkSpaces that have a user connected.	DirectoryID WorkspaceID	Average, Sum, Maximum, Minimum, Data Samples	Count
Stopped	The number of WorkSpaces that are stopped.	DirectoryID WorkspaceID	Average, Sum, Maximum, Minimum, Data Samples	Count
Maintenance[4]	The number of WorkSpaces that are under maintenance.	DirectoryID WorkspaceID	Average, Sum, Maximum, Minimum, Data Samples	Count

[1] Amazon WorkSpaces periodically sends status requests to a WorkSpace. A WorkSpace is marked `Available` when it responds to these requests, and `Unhealthy` when it fails to respond to these requests. These metrics are available at a per-WorkSpace granularity, and also aggregated for all WorkSpaces in an organization.

[2] Amazon WorkSpaces records metrics on connections made to each WorkSpace. These metrics are emitted after a user has successfully authenticated via the WorkSpaces client and the client then initiates a session. The metrics are available at a per-WorkSpace granularity, and also aggregated for all WorkSpaces in a directory.

[3] Amazon WorkSpaces periodically sends connection status requests to a WorkSpace. Users are reported as connected when they are actively using their sessions. This metric is available at a per-WorkSpace granularity, and is also aggregated for all WorkSpaces in an organization.

[4] This metric applies to WorkSpaces that are configured with an AutoStop running mode. If you have maintenance enabled for your WorkSpaces, this metric captures the number of WorkSpaces that are

currently under maintenance. This metric is available at a per-WorkSpace granularity, which describes when a WorkSpace went into maintenance and when it was removed.

Dimensions for Amazon WorkSpaces Metrics

Amazon WorkSpaces metrics are available for the following dimensions.

Dimension	Description
`DirectoryId`	Limits the data you receive to the WorkSpaces in the specified directory. The `DirectoryId` value is in the form of `d-XXXXXXXXXX`.
`WorkspaceId`	Limits the data you receive to the specified WorkSpace. The `WorkspaceId` value is in the form `ws-XXXXXXXXXX`.

Creating Amazon CloudWatch Alarms

You can create a CloudWatch alarm that watches a single metric. The alarm performs one or more actions based on the value of the metric relative to a threshold over a number of time periods. The action can be an Amazon EC2 action, an Amazon EC2 Auto Scaling action, or a notification sent to an Amazon SNS topic. You can also add alarms to CloudWatch dashboards and monitor them visually. When an alarm is on a dashboard, it turns red when it is in the `ALARM` state, making it easier for you to monitor its status proactively.

Alarms invoke actions for sustained state changes only. CloudWatch alarms do not invoke actions simply because they are in a particular state, the state must have changed and been maintained for a specified number of periods.

After an alarm invokes an action due to a change in state, its subsequent behavior depends on the type of action that you have associated with the alarm. For Amazon EC2 Auto Scaling actions, the alarm continues to invoke the action for every period that the alarm remains in the new state. For Amazon SNS notifications, no additional actions are invoked.

> **Note**
> CloudWatch doesn't test or validate the actions that you specify, nor does it detect any Amazon EC2 Auto Scaling or Amazon SNS errors resulting from an attempt to invoke nonexistent actions. Make sure that your actions exist.

Alarm States

An alarm has the following possible states:

- `OK`—The metric is within the defined threshold.
- `ALARM`—The metric is outside of the defined threshold.
- `INSUFFICIENT_DATA`—The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

Evaluating an Alarm

When you create an alarm, you specify three settings to enable CloudWatch to evaluate when to change the alarm state:

- **Period** is the length of time to evaluate the metric to create each individual data point for a metric. It is expressed in seconds. If you choose one minute as the period, there is one datapoint every minute.
- **Evaluation Period** is the number of the most recent data points to evaluate when determining alarm state.
- **Datapoints to Alarm** is the number of data points within the evaluation period that must be breaching to cause the alarm to go to the `ALARM` state. The breaching data points do not have to be consecutive, they just must all be within the last number of data points equal to **Evaluation Period**.

In the following figure, the alarm threshold is set to three units. The alarm is configured to go to the `ALARM` state and both **Evaluation Period** and **Datapoints to Alarm** are 3. That is, when all three datapoints in the most recent three consecutive periods are above the threshold, the alarm goes to the

`ALARM` state. In the figure, this happens in the third through fifth time periods. At period six, the value dips below the threshold, so one of the periods being evaluated is not breaching, and the alarm state changes to `OK`. During the ninth time period, the threshold is breached again, but for only one period. Consequently, the alarm state remains `OK`.
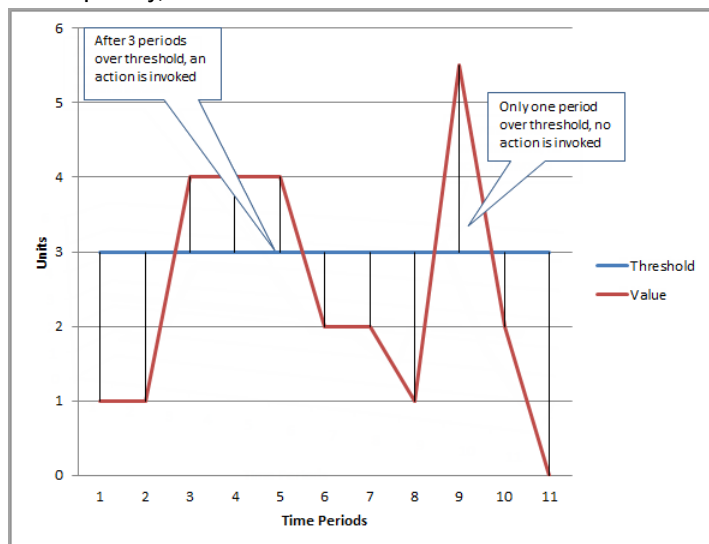
When you configure **Evaluation Period** and **Datapoints to Alarm** as different values, you are setting an "M out of N" alarm. **Datapoints to Alarm** is ("M") and **Evaluation Period** is ("N"). The evaluation interval is the number of datapoints multiplied by the period. For example, if you configure 4 out of 5 datapoints with a period of 1 minute, the evaluation interval is 5 minutes. If you configure 3 out of 3 datapoints with a period of 10 minutes, the evaluation interval is 30 minutes.

Configuring How CloudWatch Alarms Treat Missing Data

Sometimes some data points for a metric with an alarm do not get reported to CloudWatch. For example, this can happen when a connection is lost, a server goes down, or when a metric reports data only intermittently by design.

CloudWatch enables you to specify how to treat missing data points when evaluating an alarm. This can help you configure your alarm to go to the `ALARM` state when appropriate for the type of data being monitored. You can avoid false positives when missing data does not indicate a problem.

Similar to how each alarm is always in one of three states, each specific data point reported to CloudWatch falls under one of three categories:

- Not breaching (within the threshold)
- Breaching (violating the threshold)
- Missing

For each alarm, you can specify CloudWatch to treat missing data points as any of the following:

- `missing`—The alarm does not consider missing data points when evaluating whether to change state
- `notBreaching`—Missing data points are treated as being within the threshold
- `breaching`—Missing data points are treated as breaching the threshold
- `ignore`—The current alarm state is maintained

The best choice depends on the type of metric. For a metric that continually reports data, such as `CPUUtilization` of an instance, you might want to treat missing data points as `breaching`, because they may indicate thatsomething is wrong. But for a metric that generates data points only when an error occurs, such as `ThrottledRequests` in Amazon DynamoDB, you would want to treat missing data as `notBreaching`. The default behavior is `missing`.

Choosing the best option for your alarm prevents unnecessary and misleading alarm condition changes, and also more accurately indicates the health of your system.

How Alarm State is Evaluated When Data is Missing

No matter what value you set for how to treat missing data, when an alarm evaluates whether to change state, CloudWatch attempts to retrieve a higher number of data points than specified by **Evaluation Periods**. The exact number of data points it attempts to retrieve depends on the length of the alarm period and whether it is based on a metric with standard resolution or high resolution. The timeframe of the data points that it attempts to retrieve is the *evaluation range*.

Once CloudWatch retrieves these data points, the following happens:

- If no data points in the evaluation range are missing, CloudWatch evaluates the alarm based on the most recent data points collected.
- If some data points in the evaluation range are missing, but the number of existing data points retrieved is equal to or more than the alarm's **Evaluation Periods**, CloudWatch evaluates the alarm state based on the most recent existing data points that were successfully retrieved. In this case, the value you set for how to treat missing data is not needed and is ignored.
- If some data points in the evaluation range are missing, and the number of existing data points that were retrieved is lower than the alarm's number of evaluation periods, CloudWatch fills in the missing data points with the result you specified for how to treat missing data, and then evaluates the alarm. However, any real data points in the evaluation range, no matter when they were reported, are included in the evaluation. CloudWatch uses missing data points only as few times as possible.

In all of these situations, the number of datapoints evaluated is equal to the value of **Evaluation Periods**. If fewer than the value of **Datapoints to Alarm** are breaching, the alarm state is set to OK. Otherwise, the state is set to ALARM.

> **Note**
> A particular case of this behavior is that CloudWatch alarms may repeatedly re-evaluate the last set of data points for a period of time after the metric has stopped flowing. This re-evaluation may cause the alarm to change state and re-execute actions, if it had changed state immediately prior to the metric stream stopping. To mitigate this behavior, use shorter periods.

The following tables illustrate examples of the alarm evaluation behavior. In the first table, **Datapoints to Alarm** and **Evaluation Periods** are both 3. CloudWatch retrieves the 5 most recent data points when evaluating the alarm.

Column 2 shows how many of these 5 data points are missing and may need to be filled in using the setting for how to treat missing data. Columns 3-6 show the alarm state that would be set for each setting of how missing data should be treated, shown at the top of each column. In the data points column, 0 is a non-breaching data point, X is a breaching data point, and - is a missing data point.

Data points	# of missing data points	MISSING	IGNORE	BREACHING	NOT BREACHING
0 - X - X	0	OK	OK	OK	OK
0 - - - -	2	OK	OK	OK	OK

Data points	# of missing data points	MISSING	IGNORE	BREACHING	NOT BREACHING
- - - - -	3	Insufficient data	Retain current state	ALARM	OK
0 X X - X	0	ALARM	ALARM	ALARM	ALARM
- - - - X	2	Insufficient data	Retain current state	ALARM	OK

In the second row of the preceding table, the alarm stays OK even if missing data is treated as breaching, because the one existing data point is not breaching, and this is evaluated along with two missing data points which are treated as breaching. The next time this alarm is evaluated, if the data is still missing it will go to ALARM, as that non-breaching data point will no longer be among the 5 most recent data points retrieved. In the fourth row, the alarm goes to ALARM state in all cases because there are enough real data points so that the setting for how to treat missing data does not need to be considered.

In the next table, the **Period** is again set to 5 minutes, and **Datapoints to Alarm** is only 2 while **Evaluation Periods** is 3. This is a 2 out of 3, M out of N alarm.

Data points	# of missing data points	MISSING	IGNORE	BREACHING	NOT BREACHING
0 - X - X	0	ALARM	ALARM	ALARM	ALARM
0 0 X 0 X	0	ALARM	ALARM	ALARM	ALARM
0 - X - -	1	OK	OK	ALARM	OK
- - - - 0	2	OK	OK	ALARM	OK
- - - - X	2	Insufficient data	Retain current state	ALARM	OK

If data points are missing soon after you create an alarm, and the metric was being reported to CloudWatch before you created the alarm, CloudWatch retrieves the most recent data points from before the alarm was created when evaluating the alarm.

High-Resolution Alarms

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. There is a higher charge for high-resolution alarms. For more information about high-resolution metrics, see Publish Custom Metrics (p. 45).

Percentile-Based CloudWatch Alarms and Low Data Samples

When you set a percentile as the statistic for an alarm, you can specify what to do when there is not enough data for a good statistical assessment. You can choose to have the alarm evaluate the statistic

anyway and possibly change the alarm state. Or, you can have the alarm ignore the metric while the sample size is low, and wait to evaluate it until there is enough data to be statistically significant.

For percentiles between 0.5 and 1.00, this setting is used when there are fewer than 10/(1-percentile) data points during the evaluation period. For example, this setting would be used if there were fewer than 1000 samples for an alarm on a p99 percentile. For percentiles between 0 and 0.5, the setting is used when there are fewer than 10/percentile data points.

Common Features of CloudWatch Alarms

The following features apply to all CloudWatch alarms:

- You can create up to 5000 alarms per region per AWS account. To create or update an alarm, you use the `PutMetricAlarm` API action (`mon-put-metric-alarm` command).

- Alarm names must contain only ASCII characters.

- You can list any or all of the currently configured alarms, and list any alarms in a particular state using `DescribeAlarms` (`mon-describe-alarms`). You can further filter the list by time range.

- You can disable and enable alarms by using `DisableAlarmActions` and `EnableAlarmActions` (`mon-disable-alarm-actions` and `mon-enable-alarm-actions`).

- You can test an alarm by setting it to any state using `SetAlarmState` (`mon-set-alarm-state`). This temporary state change lasts only until the next alarm comparison occurs.

- You can create an alarm using `PutMetricAlarm` (`mon-put-metric-alarm`) before you've created a custom metric. For the alarm to be valid, you must include all of the dimensions for the custom metric in addition to the metric namespace and metric name in the alarm definition.

- You can view an alarm's history using `DescribeAlarmHistory` (`mon-describe-alarm-history`). CloudWatch preserves alarm history for two weeks. Each state transition is marked with a unique time stamp. In rare cases, your history might show more than one notification for a state change. The time stamp enables you to confirm unique state changes.

- The number of evaluation periods for an alarm multiplied by the length of each evaluation period cannot exceed one day.

> **Note**
> Some AWS resources do not send metric data to CloudWatch under certain conditions.
> For example, Amazon EBS may not send metric data for an available volume that is not attached to an Amazon EC2 instance, because there is no metric activity to be monitored for that volume. If you have an alarm set for such a metric, you may notice its state change to Insufficient Data. This may indicate that your resource is inactive, and may not necessarily mean that there is a problem.

Set Up Amazon SNS Notifications

Amazon CloudWatch uses Amazon SNS to send email. First, create and subscribe to an SNS topic. When you create a CloudWatch alarm, you can add this SNS topic to send an email notification when the alarm changes state. For more information, see the Amazon Simple Notification Service Getting Started Guide.

> **Note**
> Alternatively, if you plan to create your CloudWatch alarm using the AWS Management Console, you can skip this procedure because you can create the topic through the **Create Alarm Wizard**.

Set Up an Amazon SNS Topic Using the AWS Management Console

First, create a topic, then subscribe to it. You can optionally publish a test message to the topic.

To create an SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.
2. On the Amazon SNS dashboard, under **Common actions**, choose **Create Topic**.
3. In the **Create new topic** dialog box, for **Topic name**, type a name for the topic (for example, my-topic).
4. Choose **Create topic**.
5. Copy the **Topic ARN** for the next task (for example, arn:aws:sns:us-east-1:111122223333:my-topic).

To subscribe to an SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.
2. In the navigation pane, choose **Subscriptions**, **Create subscription**.
3. In the **Create subscription** dialog box, for **Topic ARN**, paste the topic ARN that you created in the previous task.
4. For **Protocol**, choose **Email**.
5. For **Endpoint**, type an email address that you can use to receive the notification, and then choose **Create subscription**.
6. From your email application and open the message from AWS Notifications and confirm your subscription.

    Your web browser displays a confirmation response from Amazon SNS.

To publish a test message to an SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.
2. In the navigation pane, choose **Topics**.
3. On the **Topics** page, select a topic and choose **Publish to topic**.
4. In the **Publish a message** page, for **Subject**, type a subject line for your message, and for **Message**, type a brief message.
5. Choose **Publish Message**.
6. Check your email to confirm that you received the message.

Set Up an SNS Topic Using the AWS CLI

First you create an SNS topic, and then publish a message directly to the topic to test that you have properly configured it.

To set up an SNS topic

1. Create the topic using the create-topic command as follows.

    ```
    aws sns create-topic --name my-topic
    ```

Amazon SNS returns a topic ARN with the following format:

```
{
    "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic"
}
```

2. Subscribe your email address to the topic using the subscribe command. If the subscription request succeeds, you receive a confirmation email message.

```
aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:my-topic --protocol
 email --notification-endpoint my-email-address
```

Amazon SNS returns the following:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

3. From your email application and open the message from AWS Notifications and confirm your subscription.

Your web browser displays a confirmation response from Amazon Simple Notification Service.

4. Check the subscription using the list-subscriptions-by-topic command.

```
aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-east-1:111122223333:my-
topic
```

Amazon SNS returns the following:

```
{
  "Subscriptions": [
    {
        "Owner": "111122223333",
        "Endpoint": "me@mycompany.com",
        "Protocol": "email",
        "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic",
        "SubscriptionArn": "arn:aws:sns:us-east-1:111122223333:my-topic:64886986-
bf10-48fb-a2f1-dab033aa67a3"
    }
  ]
}
```

5. (Optional) Publish a test message to the topic using the publish command.

```
aws sns publish --message "Verification" --topic arn:aws:sns:us-east-1:111122223333:my-
topic
```

Amazon SNS returns the following:

```
{
    "MessageId": "42f189a0-3094-5cf6-8fd7-c2dde61a4d7d"
}
```

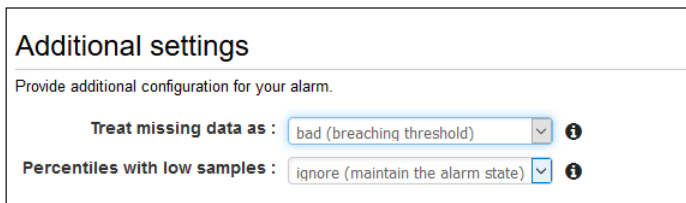6. Check your email to confirm that you received the message.

Create or Edit a CloudWatch Alarm

You can choose specific metrics to trigger the alarm and specify thresholds for those metrics. You can then set your alarm to change state when a metric exceeds a threshold that you have defined.

To create an alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms**, **Create Alarm**.

3. For the **Select Metric** step, do the following:

   a. Choose a metric category (for example, **EC2 Metrics**).

   b. Select an instance and metric (for example, **CPUUtilization**).

   c. For the statistic, choose one of the statistics (for example, Average) or predefined percentiles, or specify a custom percentile (for example, p95.45).

   d. Choose a period (for example, `1 Hour`).

   e. Choose **Next**.

4. For the **Define Alarm** step, do the following:

   a. Under **Alarm Threshold**, type a unique name for the alarm and a description of the alarm. The alarm name must contain only ASCII characters. For **Whenever**, specify a threshold (for example, 80 percent of CPU utilization) and the number of datapoints ("M" out of "N") that must be breaching to trigger the alarm. For more information, see Evaluating an Alarm (p. 327).

   b. Under **Additional settings**, for **Treat missing data as**, choose how to have the alarm treat missing data points. For more information, see Configuring How CloudWatch Alarms Treat Missing Data (p. 328).

   If the alarm uses a percentile as the monitored statistic, choose whether to evaluate or ignore cases with low sample rates. If you choose **ignore**, the current alarm state is maintained when the sample size is too low. For more information, see Percentile-Based CloudWatch Alarms and Low Data Samples (p. 330).

   c. Under **Actions**, select the type of action to have the alarm to perform when the alarm is triggered.

   d. Choose **Create Alarm**.

You can also add alarms to a dashboard. For more information, see Add or Remove an Alarm from a CloudWatch Dashboard (p. 26).

To edit an alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms**.

3. Select the alarm, and then choose **Actions**, **Modify**.

4. In the **Modify Alarm** dialog box, update the alarm as necessary and choose **Save Changes**.

You cannot change the name of an existing alarm. You can change the description. Or you can copy the alarm, and give the new alarm a different name. To copy an alarm, select the alarm and then choose **Actions**, **Copy**.

To update an email notification list that was created using the Amazon SNS console

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v2/home.

2. In the navigation pane, choose **Topics**, and then select the ARN for your notification list (topic).

3. Do one of the following:

   - To add an email address, choose **Create subscription**. For **Protocol**, choose **Email**. For **Endpoint**, type the email address of the new recipient. Choose **Create subscription**.

   - To remove an email address, choose the **Subscription ID**. Choose **Other subscription actions**, **Delete subscriptions**.

4. Choose **Publish to topic**.

Create a CPU Usage Alarm that Sends Email

You can create an CloudWatch alarm that sends an email message using Amazon SNS when the alarm changes state from OK to ALARM.

The alarm changes to the ALARM state when the average CPU use of an EC2 instance exceeds a specified threshold for consecutive specified periods.

Set Up a CPU Usage Alarm Using the AWS Management Console

Use these steps to use the AWS Management Console to create a CPU usage alarm.

To create an alarm that sends email based on CPU usage

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms**, **Create Alarm**.

3. Under **EC2 Metrics**, choose a metric category (for example, **Per-Instance Metrics**).

4. Select a metric as follows:

   a. Select a row with the instance and the **CPUUtilization** metric.

   b. For the statistic, choose **Average**, choose one of the predefined percentiles, or specify a custom percentile (for example, p95.45).

   c. Choose a period (for example, `5 minutes`).

   d. Choose **Next**.

5. Define the alarm as follows:

   a. Under **Alarm Threshold**, type a unique name for the alarm (for example, myHighCpuAlarm) and a description of the alarm (for example, CPU usage exceeds 70 percent). Alarm names must contain only ASCII characters.

   b. Under **Whenever**, for **is**, choose **>** and type **70**. For **for**, type **2**. This specifies that the alarm is triggered if the CPU usage is above 70 percent for two consecutive sampling periods.
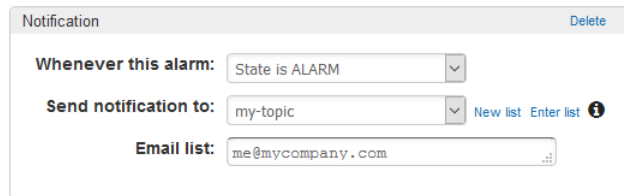

   c. Under **Additional settings**, for **Treat missing data as**, choose **bad (breaching threshold)**, as missing data points may indicate that the instance is down.

   d. Under **Actions**, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, select an existing SNS topic or create a new one.

e.   To create a new SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, myHighCpuAlarm), and for **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent.

f.   Choose **Create Alarm**.

Set Up a CPU Usage Alarm Using the AWS CLI

Use these steps to use the AWS CLI to create a CPU usage alarm.

To create an alarm that sends email based on CPU usage

1.   Set up an SNS topic. For more information, see Set Up Amazon SNS Notifications (p. 331).

2.   Create an alarm using the put-metric-alarm command as follows.

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70%" --metric-name CPUUtilization --namespace AWS/EC2 --statistic
 Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --
dimensions  Name=InstanceId,Value=i-12345678 --evaluation-periods 2 --alarm-actions
 arn:aws:sns:us-east-1:111122223333:my-topic --unit Percent
```

3.   Test the alarm by forcing an alarm state change using the set-alarm-state command.

a.   Change the alarm state from `INSUFFICIENT_DATA` to `OK`:

```
aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason "initializing"
 --state-value OK
```

b.   Change the alarm state from `OK` to `ALARM`:

```
aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason "initializing"
 --state-value ALARM
```

c.   Check that you have received an email notification about the alarm.

Create a Load Balancer Latency Alarm that Sends Email

You can set up an Amazon SNS notification and configure an alarm that monitors latency exceeding 100 ms for your Classic Load Balancer.

Set Up a Latency Alarm Using the AWS Management Console

Use these steps to use the AWS Management Console to create a load balancer latency alarm.

To create a load balancer latency alarm that sends email

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Under **CloudWatch Metrics by Category**, choose the **ELB Metrics** category.
4. Select the row with the Classic Load Balancer and the **Latency** metric.
5. For the statistic, choose **Average**, choose one of the predefined percentiles, or specify a custom percentile (for example, p95.45).
6. For the period, choose **1 Minute**.
7. Choose **Next**.
8. Under **Alarm Threshold**, type a unique name for the alarm (for example, `myHighCpuAlarm`) and a description of the alarm (for example, Alarm when Latency exceeds 100s). Alarm names must contain only ASCII characters.
9. Under **Whenever**, for **is**, choose **>** and type **0.1**. For **for**, type **3**.
10. Under **Additional settings**, for **Treat missing data as**, choose **ignore (maintain alarm state)** so that missing data points do not trigger alarm state changes.

    For **Percentiles with low samples** choose **ignore (maintain the alarm state)** so that the alarm evaluates only situations with adequate numbers of data samples.
11. Under **Actions**, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to** choose an existing SNS topic or create a new one.

    To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, `myHighCpuAlarm`), and for **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent.
12. Choose **Create Alarm**.

Set Up a Latency Alarm Using the AWS CLI

Use these steps to use the AWS CLI to create a load balancer latency alarm.

To create a load balancer latency alarm that sends email

1. Set up an SNS topic. For more information, see Set Up Amazon SNS Notifications (p. 331).
2. Create the alarm using the put-metric-alarm command as follows:

```
aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm
 when Latency exceeds 100s" --metric-name Latency --namespace AWS/ELB --statistic
 Average --period 60 --threshold 100 --comparison-operator GreaterThanThreshold --
dimensions Name=LoadBalancerName,Value=my-server --evaluation-periods 3 --alarm-actions
 arn:aws:sns:us-east-1:111122223333:my-topic --unit Seconds
```

3. Test the alarm by forcing an alarm state change using the set-alarm-state command.

    a. Change the alarm state from `INSUFFICIENT_DATA` to `OK`:

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --
state-value OK
```

b. Change the alarm state from `OK` to `ALARM`:

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --
state-value ALARM
```

c. Check that you have received an email notification about the alarm.

Create a Storage Throughput Alarm that Sends Email

You can set up an SNS notification and configure an alarm that sends email when Amazon EBS exceeds 100MB throughput.

Set Up a Storage Throughput Alarm Using the AWS Management Console

Use these steps to use the AWS Management Console to create an alarm based on Amazon EBS throughput.

To create a storage throughput alarm that sends email

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms**, **Create Alarm**.

3. Under **EBS Metrics**, choose a metric category.

4. Select the row with the volume and the **VolumeWriteBytes** metric.

5. For the statistic, choose **Average**. For the period, choose **5 Minutes**. Choose **Next**.

6. Under **Alarm Threshold**, type a unique name for the alarm (for example, myHighWriteAlarm) and a description of the alarm (for example, VolumeWriteBytes exceeds 100,000 KiB/s). Alarm names must contain only ASCII characters.

7. Under **Whenever**, for **is**, choose **>** and type **100000**. For **for**, type **15** consecutive periods.

    A graphical representation of the threshold is shown under **Alarm Preview**.

8. Under **Additional settings**, for **Treat missing data as**, choose **ignore (maintain alarm state)** so that missing data points do not trigger alarm state changes.

9. Under **Actions**, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose an existing SNS topic or create one.

    To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, myHighCpuAlarm), and for **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

10. Choose **Create Alarm**.

Set Up a Storage Throughput Alarm Using the AWS CLI

Use these steps to use the AWS CLI to create an alarm based on Amazon EBS throughput.

To create a storage throughput alarm that sends email

1. Create an SNS topic. For more information, see .

2. Create the alarm.

   ```
   aws cloudwatch put-metric-alarm --alarm-name ebs-mon --alarm-description "Alarm when
    EBS volume exceeds 100MB throughput" --metric-name VolumeReadBytes --namespace AWS/
   EBS --statistic Average --period 300 --threshold 100000000 --comparison-operator
    GreaterThanThreshold --dimensions Name=VolumeId,Value=my-volume-id --evaluation-
   periods 3 --alarm-actions arn:aws:sns:us-east-1:111122223333:my-alarm-topic --
   insufficient-data-actions arn:aws:sns:us-east-1:111122223333:my-insufficient-data-topic
   ```

3. Test the alarm by forcing an alarm state change using the set-alarm-state command.

   a. Change the alarm state from `INSUFFICIENT_DATA` to `OK`:

      ```
      aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing"
       --state-value OK
      ```

   b. Change the alarm state from `OK` to `ALARM`:

      ```
      aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing"
       --state-value ALARM
      ```

   c. Change the alarm state from `ALARM` to `INSUFFICIENT_DATA`:

      ```
      aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing"
       --state-value INSUFFICIENT_DATA
      ```

   d. Check that you have received an email notification about the alarm.

Create Alarms to Stop, Terminate, Reboot, or Recover an Instance

Using Amazon CloudWatch alarm actions, you can create alarms that automatically stop, terminate, reboot, or recover your EC2 instances. You can use the stop or terminate actions to help you save money when you no longer need an instance to be running. You can use the reboot and recover actions to automatically reboot those instances or recover them onto new hardware if a system impairment occurs.

There are a number of scenarios in which you might want to automatically stop or terminate your instance. For example, you might have instances dedicated to batch payroll processing jobs or scientific computing tasks that run for a period of time and then complete their work. Rather than letting those instances sit idle (and accrue charges), you can stop or terminate them which can help you to save money. The main difference between using the stop and the terminate alarm actions is that you can easily restart a stopped instance if you need to run it again later, and you can keep the same instance ID and root volume. However, you cannot restart a terminated instance. Instead, you must launch a new instance.

You can add the stop, terminate, reboot, or recover actions to any alarm that is set on an Amazon EC2 per-instance metric, including basic and detailed monitoring metrics provided by Amazon CloudWatch (in the AWS/EC2 namespace), in addition to any custom metrics that include the "InstanceId=" dimension, as long as the InstanceId value refers to a valid running Amazon EC2 instance.

To set up a CloudWatch alarm action that can reboot, stop, or terminate an instance, you must use a service-linked IAM role, *AWSServiceRoleForCloudWatchEvents*. The AWSServiceRoleForCloudWatchEvents IAM role enables AWS to perform alarm actions on your behalf.

To create the service-linked role for CloudWatch Events, use the following command:

```
aws iam create-service-linked-role --aws-service-name events.amazonaws.com
```

Console Support

You can create alarms using the CloudWatch console or the Amazon EC2 console. The procedures in this documentation use the CloudWatch console. For procedures that use the Amazon EC2 console, see Create Alarms That Stop, Terminate, Reboot, or Recover an Instance in the *Amazon EC2 User Guide for Linux Instances*.

Permissions

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following permissions:

- `iam:CreateServiceLinkedRole`, `iam:GetPolicy`, `iam:GetPolicyVersion`, and `iam:GetRole` — For all alarms with Amazon EC2 actions
- `ec2:DescribeInstanceStatus` and `ec2:DescribeInstances` — For all alarms on Amazon EC2 instance status metrics
- `ec2:StopInstances` — For alarms with stop actions
- `ec2:TerminateInstances` — For alarms with terminate actions
- No specific permissions are needed for alarms with recover actions

If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or terminate actions won't be performed on the instance. However, if you are later granted permission to use the associated Amazon EC2 API actions, the alarm actions you created earlier will be performed. For more information, see Permissions and Policies in the *IAM User Guide*.

If you want to use an IAM role to stop, terminate, or reboot an instance using an alarm action, you can only use the AWSServiceRoleForCloudWatchEvents role. Other IAM roles are not supported. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Amazon EC2 Auto Scaling policies.

Contents
- Adding Stop Actions to Amazon CloudWatch Alarms (p. 341)
- Adding Terminate Actions to Amazon CloudWatch Alarms (p. 342)
- Adding Reboot Actions to Amazon CloudWatch Alarms (p. 343)
- Adding Recover Actions to Amazon CloudWatch Alarms (p. 344)
- Viewing the History of Triggered Alarms and Actions (p. 345)

Adding Stop Actions to Amazon CloudWatch Alarms

You can create an alarm that stops an Amazon EC2 instance when a certain threshold has been met. For example, you may run development or test instances and occasionally forget to shut them off. You can

create an alarm that is triggered when the average CPU utilization percentage has been lower than 10 percent for 24 hours, signaling that it is idle and no longer in use. You can adjust the threshold, duration, and period to suit your needs, plus you can add an SNS notification, so that you will receive an email when the alarm is triggered.

Amazon EC2 instances that use an Amazon Elastic Block Store volume as the root device can be stopped or terminated, whereas instances that use the instance store as the root device can only be terminated.

To create an alarm to stop an idle instance using the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. For the **Select Metric** step, do the following:

    a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.
    b. Select the row with the instance and the **CPUUtilization** metric.
    c. For the statistic, choose **Average**.
    d. Choose a period (for example, `1 Hour`).
    e. Choose **Next**.
4. For the **Define Alarm** step, do the following:

    a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Stop EC2 instance) and a description of the alarm (for example, Stop EC2 instance when CPU is idle too long). Alarm names must contain only ASCII characters.
    b. Under **Whenever**, for **is**, choose **<** and type `10`. For **for**, type `24` consecutive periods.

        A graphical representation of the threshold is shown under **Alarm Preview**.
    c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

        To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Stop_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.
    d. Choose **EC2 Action**.
    e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Stop this instance**.
    f. Choose **Create Alarm**.

Adding Terminate Actions to Amazon CloudWatch Alarms

You can create an alarm that terminates an EC2 instance automatically when a certain threshold has been met (as long as termination protection is not enabled for the instance). For example, you might want to terminate an instance when it has completed its work, and you don't need the instance again. If you might want to use the instance later, you should stop the instance instead of terminating it. For information about enabling and disabling termination protection for an instance, see Enabling Termination Protection for an Instance in the *Amazon EC2 User Guide for Linux Instances*.

To create an alarm to terminate an idle instance using the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.

3. For the **Select Metric** step, do the following:

   a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.

   b. Select the row with the instance and the **CPUUtilization** metric.

   c. For the statistic, choose **Average**.

   d. Choose a period (for example, `1 Hour`).

   e. Choose **Next**.

4. For the **Define Alarm** step, do the following:

   a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Terminate EC2 instance) and a description of the alarm (for example, Terminate EC2 instance when CPU is idle for too long). Alarm names must contain only ASCII characters.

   b. Under **Whenever**, for **is**, choose **<** and type **10**. For **for**, type **24** consecutive periods.

      A graphical representation of the threshold is shown under **Alarm Preview**.

   c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

      To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Terminate_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

   d. Choose **EC2 Action**.

   e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Terminate this instance**.

   f. Choose **Create Alarm**.

Adding Reboot Actions to Amazon CloudWatch Alarms

You can create an Amazon CloudWatch alarm that monitors an Amazon EC2 instance and automatically reboots the instance. The reboot alarm action is recommended for Instance Health Check failures (as opposed to the recover alarm action, which is suited for System Health Check failures). An instance reboot is equivalent to an operating system reboot. In most cases, it takes only a few minutes to reboot your instance. When you reboot an instance, it remains on the same physical host, so your instance keeps its public DNS name, private IP address, and any data on its instance store volumes.

Rebooting an instance doesn't start a new instance billing hour, unlike stopping and restarting your instance. For more information about rebooting an instance, see Reboot Your Instance in the *Amazon EC2 User Guide for Linux Instances*.

> **Important**
> To avoid a race condition between the reboot and recover actions, avoid setting the same evaluation period for both a reboot alarm and a recover alarm. We recommend that you set reboot alarms to three evaluation periods of one minute each.

To create an alarm to reboot an instance using the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms**, **Create Alarm**.

3. For the **Select Metric** step, do the following:

   a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.

    b.   Select the row with the instance and the **StatusCheckFailed_Instance** metric.

    c.   For the statistic, choose **Minimum**.

    d.   Choose a period (for example, `1 Minute`) and choose **Next**.

4.   For the **Define Alarm** step, do the following:

    a.   Under **Alarm Threshold**, type a unique name for the alarm (for example, Reboot EC2 instance) and a description of the alarm (for example, Reboot EC2 instance when health checks fail). Alarm names must contain only ASCII characters.

    b.   Under **Whenever**, for **is**, choose **>** and type `0`. For **for**, type `3` consecutive periods.

        A graphical representation of the threshold is shown under **Alarm Preview**.

    c.   Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

        To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Reboot_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

    d.   Choose **EC2 Action**.

    e.   For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Reboot this instance**.

    f.   Choose **Create Alarm**.

Adding Recover Actions to Amazon CloudWatch Alarms

You can create an Amazon CloudWatch alarm that monitors an Amazon EC2 instance and automatically recovers the instance if it becomes impaired due to an underlying hardware failure or a problem that requires AWS involvement to repair. Terminated instances cannot be recovered. A recovered instance is identical to the original instance, including the instance ID, private IP addresses, Elastic IP addresses, and all instance metadata.

When the `StatusCheckFailed_System` alarm is triggered, and the recover action is initiated, you will be notified by the Amazon SNS topic that you chose when you created the alarm and associated the recover action. During instance recovery, the instance is migrated during an instance reboot, and any data that is in-memory is lost. When the process is complete, information is published to the SNS topic you've configured for the alarm. Anyone who is subscribed to this SNS topic will receive an email notification that includes the status of the recovery attempt and any further instructions. You will notice an instance reboot on the recovered instance.

The recover action can be used only with `StatusCheckFailed_System`, not with `StatusCheckFailed_Instance`.

Examples of problems that cause system status checks to fail include:

- Loss of network connectivity
- Loss of system power
- Software issues on the physical host
- Hardware issues on the physical host that impact network reachability

The recover action is only supported on:

- The C3, C4, C5, M3, M4, M5, R3, R4, T2, and X1 instance types

- Instances in a VPC
- Instances with `default` or `dedicated` instance tenancy
- Instances that use Amazon EBS volumes only (do not configure instance store volumes)

If your instance has a public IPv4 address, it retains the public IP address after recovery.

Important
To avoid a race condition between the reboot and recover actions, avoid setting the same evaluation period for both a reboot alarm and a recover alarm. We recommend that you set recover alarms to two evaluation periods of one minute each and reboot alarms to three evaluation periods of one minute each.

To create an alarm to recover an instance using the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. For the **Select Metric** step, do the following:

    a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.
    b. Select the row with the instance and the **StatusCheckFailed_System** metric.
    c. For the statistic, choose **Minimum**.
    d. Choose a period (for example, `1 Minute`).

        Important
        To avoid a race condition between the reboot and recover actions, avoid setting the same evaluation period for both a reboot alarm and a recover alarm. We recommend that you set recover alarms to two evaluation periods of one minute each.

    e. Choose **Next**.
4. For the **Define Alarm** step, do the following:

    a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Recover EC2 instance) and a description of the alarm (for example, Recover EC2 instance when health checks fail). Alarm names must contain only ASCII characters.
    b. Under **Whenever**, for **is**, choose **>** and type `0`. For **for**, type `2` consecutive periods.
    c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

        To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Recover_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.
    d. Choose **EC2 Action**.
    e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Recover this instance**.
    f. Choose **Create Alarm**.

Viewing the History of Triggered Alarms and Actions

You can view alarm and action history in the Amazon CloudWatch console. Amazon CloudWatch keeps the last two weeks' worth of alarm and action history.

To view the history of triggered alarms and actions

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Alarms** and select an alarm.

3. To view the most recent state transition along with the time and metric values, choose **Details**.

4. To view the most recent history entries, choose **History**.

Create a Billing Alarm to Monitor Your Estimated AWS Charges

You can monitor your estimated AWS charges using Amazon CloudWatch. When you enable the monitoring of estimated charges for your AWS account, the estimated charges are calculated and sent several times daily to CloudWatch as metric data.

Billing metric data is stored in the US East (N. Virginia) region and represents worldwide charges. This data includes the estimated charges for every service in AWS that you use, in addition to the estimated overall total of your AWS charges.

The alarm triggers when your account billing exceeds the threshold you specify. It triggers only when actual billing exceeds the threshold. It does not use projections based on your usage so far in the month.

If you create a billing alarm at a time when your charges have already exceeded the threshold, the alarm goes to the `ALARM` state immediately.

Tasks

- Enable Billing Alerts (p. 346)
- Create a Billing Alarm (p. 347)
- Check the Alarm Status (p. 348)
- Delete a Billing Alarm (p. 348)

Enable Billing Alerts

Before you can create an alarm for your estimated charges, you must enable billing alerts, so that you can monitor your estimated AWS charges and create an alarm using billing metric data. After you enable billing alerts, you cannot disable data collection, but you can delete any billing alarms that you created.

After you enable billing alerts for the first time, it takes about 15 minutes before you can view billing data and set billing alarms.

Requirements

- You must be signed in using AWS account root user credentials; IAM users cannot enable billing alerts for your AWS account.
- For consolidated billing accounts, billing data for each linked account can be found by logging in as the paying account. You can view billing data for total estimated charges and estimated charges by service for each linked account, in addition to the consolidated account.

To enable the monitoring of estimated charges

1. Open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home?#.

2. In the navigation pane, choose **Preferences**.

3. Choose **Receive Billing Alerts**.

4.   Choose **Save preferences**.

Create a Billing Alarm

After you've enabled billing alerts, you can create a billing alarm. In this procedure, you create an alarm that sends an email message when your estimated charges for AWS exceed a specified threshold.

> **Note**
> This procedure uses the advanced options. For more information about using the simple options, see Create a Billing Alarm (p. 14) in *Monitor Your Estimated Charges Using CloudWatch*.

To create a billing alarm using the CloudWatch console

1.   Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2.   If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and represents worldwide charges.
3.   In the navigation pane, choose **Alarms**, **Billing**, **Create Alarm**.
4.   Choose **show advanced** to switch to the advanced options.
5.   Under **Alarm Threshold**, replace the default name for the alarm (for example, My Estimated Charges) and a description for the alarm (for example, Estimated Monthly Charges). Alarm names must contain only ASCII characters.
6.   Under **Whenever charges for**, for **is**, choose **>=** and then type the monetary amount (for example, 200) that must be exceeded to trigger the alarm and send an email.

> **Note**
> Under **Alarm Preview**, there is an estimate of your charges that you can use to set an appropriate amount.

7.   Under **Additional settings**, for **Treat missing data as**, choose **ignore (maintain alarm state)** so that missing data points do not trigger alarm state changes.
8.   Under **Actions**, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose an existing SNS topic or create a new one.

To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic, and for **Email list**, type a comma-separated list of email addresses where email notifications should be sent. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

9. Choose **Create Alarm**.

Check the Alarm Status

You can check the status of your billing alarm.

To check alarm status

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**, **Billing**.
4. Select the check box next to the alarm. Until the subscription is confirmed, it is shown as "Pending confirmation". After the subscription is confirmed, refresh the console to show the updated status.

Delete a Billing Alarm

You can delete your billing alarm when you no longer need it.

To delete a billing alarm

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. If necessary, change the region to US East (N. Virginia). Billing metric data is stored in this region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**, **Billing**.
4. Select the check box next to the alarm and choose **Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Hide Amazon EC2 Auto Scaling Alarms

When you view your alarms in the AWS Management Console, you can hide the alarms related to Amazon EC2 Auto Scaling. This feature is available only in the AWS Management Console.

To temporarily hide Amazon EC2 Auto Scaling alarms

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms** and select **Hide all AutoScaling alarms**.

Using CloudWatch with Interface VPC Endpoints

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and CloudWatch. You can use this connection to enable CloudWatch to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such the IP address range, subnets, route tables, and network gateways. To connect your VPC to CloudWatch, you define an *interface VPC endpoint* for CloudWatch. This type of endpoint enables you to connect your VPC to AWS services. The endpoint provides reliable, scalable connectivity to CloudWatch without requiring an internet gateway, network address translation (NAT) instance, or VPN connection. For more information, see What is Amazon VPC in the *Amazon VPC User Guide*.

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see New – AWS PrivateLink for AWS Services.

The following steps are for users of Amazon VPC. For more information, see Getting Started in the *Amazon VPC User Guide*.

Availability

CloudWatch currently supports VPC endpoints in the following Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- EU (Frankfurt)
- EU (Ireland)
- EU (London)
- EU (Paris)
- South America (São Paulo)

Create a VPC Endpoint for CloudWatch

To start using CloudWatch with your VPC, create an interface VPC endpoint for CloudWatch. For more information, see Creating an Interface Endpoint in the *Amazon VPC User Guide*.

You do not need to change the settings for CloudWatch. CloudWatch calls other AWS services using either public endpoints or private interface VPC endpoints, whichever are in use. For example, if you create an interface VPC endpoint for CloudWatch, and you already have a metrics flowing to CloudWatch from resources located on your VPC, these metrics begin flowing through the interface VPC endpoint by default.

Authentication and Access Control for Amazon CloudWatch

Access to Amazon CloudWatch requires credentials. Those credentials must have permissions to access AWS resources, such as retrieving CloudWatch metric data about your cloud resources. The following sections provide details about how you can use AWS Identity and Access Management (IAM) and CloudWatch to help secure your resources by controlling who can access them:

- Authentication (p. 351)
- Access Control (p. 352)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your *AWS account user credentials* and they provide complete access to all of your AWS resources.

    Important
    For security reasons, we recommend that you use the AWS account user credentials only to create an *administrator*, which is an *IAM user* with full permissions to your AWS account. Then, you can use this administrator to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the *IAM User Guide*.

- **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to view metrics in CloudWatch). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

    In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and AWS CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. CloudWatch supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

- **IAM role** – An IAM role is another IAM identity you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

    - **Federated user access** – Instead of creating an IAM user, you can use preexisting identities from AWS Directory Service, your enterprise user directory, or a web identity provider (IdP). These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an IdP. For more information, see Federated Users and Roles in the *IAM User Guide*.

- **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM Roles in the *IAM User Guide*.

- **AWS service access** – You can use an IAM role in your account to grant an AWS service the permissions needed to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the *IAM User Guide*.

- **Applications running on Amazon EC2** – Instead of storing access keys within the EC2 instance for use by applications running on the instance and making API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access CloudWatch resources. For example, you must have permissions to create CloudWatch dashboard widgets, view metrics, and so on.

The following sections describe how to manage permissions for CloudWatch. We recommend that you read the overview first.

- Overview of Managing Access Permissions to Your CloudWatch Resources (p. 353)
- Using Identity-Based Policies (IAM Policies) for CloudWatch (p. 356)
- Amazon CloudWatch Permissions Reference (p. 365)

CloudWatch Dashboard Permissions Update

On May 1, 2018, the permissions required to access CloudWatch dashboards will change. Currently, the **cloudwatch:GetMetricData** permission is required to view CloudWatch dashboards, and the **cloudwatch:PutMetricData** permission is required to create or modify dashboards. Beginning on May 1, dashboard access in the CloudWatch console will instead require newer permissions that were introduced in 2017 to support dashboard API operations:

- **cloudwatch:GetDashboard**
- **cloudwatch:ListDashboards**
- **cloudwatch:PutDashboard**
- **cloudwatch:DeleteDashboards**

To check whether you will have access to CloudWatch dashboards after the change, choose **Check permissions** in update messages in the CloudWatch console. If that check shows that you will not have these permissions after the update, you should use the IAM console to fix your permissions before May 1.

To retain access to CloudWatch dashboards, you need one of the following:

- The **AdministratorAccess** policy.
- The **CloudWatchFullAccess** policy.
- A custom policy that includes one or more of these specific permissions:
  - `cloudwatch:GetDashboard` and `cloudwatch:ListDashboards` to be able to view dashboards
  - `cloudwatch:PutDashboard` to be able to create or modify dashboards
  - `cloudwatch:DeleteDashboards` to be able to delete dashboards

For more information for changing permissions for an IAM user using policies, see Changing Permissions for an IAM User.

For more information about CloudWatch permissions, see Amazon CloudWatch Permissions Reference (p. 365).

For more information about dashboard API operations, see PutDashboard in the Amazon CloudWatch API Reference.

Overview of Managing Access Permissions to Your CloudWatch Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

> **Note**
> An *account administrator* (or administrator IAM user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

CloudWatch Resources and Operations

CloudWatch doesn't have any specific resources for you to control access to. Therefore, there are no CloudWatch Amazon Resource Names (ARNs) for you to use in an IAM policy. For example, you can't give a user access to CloudWatch data for only a specific set of EC2 instances or a specific load balancer. Permissions granted using IAM cover all the cloud resources you use or monitor with CloudWatch. In addition, you can't use IAM roles with the CloudWatch command line tools.

You use an * (asterisk) as the resource when writing a policy to control access to CloudWatch actions. For example:

```
{
  "Version": "2012-10-17",
  "Statement":[{
      "Effect":"Allow",
      "Action":["cloudwatch:GetMetricStatistics","cloudwatch:ListMetrics"],
      "Resource":"*",
      "Condition":{
          "Bool":{
              "aws:SecureTransport":"true"
              }
          }
      }
  ]
  }
```

For more information about ARNs, see ARNs in *IAM User Guide*. For information about CloudWatch Logs ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces in the *Amazon Web Services General Reference*. For an example of a policy that covers CloudWatch actions, see Using Identity-Based Policies (IAM Policies) for CloudWatch (p. 356).

Action	ARN (with region)	ARN (for use with IAM role)
Stop	arn:aws:automate:us-east-1:ec2:stop	arn:aws:swf:us-east-1:*customer-account*:action/actions/AWS_EC2.InstanceId.Stop/1.0
Terminate	arn:aws:automate:us-east-1:ec2:terminate	arn:aws:swf:us-east-1:*customer-account*:action/actions/AWS_EC2.InstanceId.Terminate/1.0
Reboot	n/a	arn:aws:swf:us-east-1:*customer-account*:action/actions/AWS_EC2.InstanceId.Reboot/1.0
Recover	arn:aws:automate:us-east-1:ec2:recover	n/a

Understanding Resource Ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the principal entity (that is, the AWS account root user, an IAM user, or an IAM role) that authenticates the resource creation request. CloudWatch does not have any resources that you can own.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
> This section discusses using IAM in the context of CloudWatch. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies) and policies attached to a resource are referred to as resource-based policies. CloudWatch supports only identity-based policies.

Topics
- Identity-Based Policies (IAM Policies) (p. 355)
- Resource-Based Policies (IAM Policies) (p. 355)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an Amazon CloudWatch resource, such as metrics, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in account A can create a role to grant cross-account permissions to another AWS account (for example, account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in account A.
  2. Account A administrator attaches a trust policy to the role identifying account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in account B. Doing this allows users in account B to create or access resources in account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

  For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

For more information about using identity-based policies with CloudWatch, see Using Identity-Based Policies (IAM Policies) for CloudWatch (p. 356). For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Resource-Based Policies (IAM Policies)

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an Amazon S3 bucket to manage access permissions to that bucket. CloudWatch doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, and Principals

For each CloudWatch resource, the service defines a set of API operations. To grant permissions for these API operations, CloudWatch defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action in order to perform the API operation. For more information about resources and API operations, see CloudWatch Resources and Operations (p. 353) and CloudWatch Actions.

The following are the basic policy elements:

- **Resource** – Use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. CloudWatch does not have any resources for you to control using policies resources, so use

the wildcard character (*) in IAM policies. For more information, see CloudWatch Resources and Operations (p. 353).

- **Action** – Use action keywords to identify resource operations that you want to allow or deny. For example, the `cloudwatch:ListMetrics` permission allows the user permissions to perform the `ListMetrics` operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the specific action. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). CloudWatch doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see AWS IAM JSON Policy Reference in the *IAM User Guide*.

For a table showing all of the CloudWatch API actions and the resources that they apply to, see Amazon CloudWatch Permissions Reference (p. 365).

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see Condition in the *IAM User Guide*.

To express conditions, you use predefined condition keys. For a list of context keys supported by each AWS service and a list of AWS-wide policy keys, see AWS Service Actions and Condition Context Keys and Global and IAM Condition Context Keys in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for CloudWatch

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform operations on CloudWatch resources.

Important
We recommend that you first review the introductory topics that explain the basic concepts and options available to manage access to your CloudWatch resources. For more information, see Access Control (p. 352).

The sections in this topic cover the following:

- Permissions Required to Use the CloudWatch Console (p. 357)
- AWS Managed (Predefined) Policies for CloudWatch (p. 359)
- Customer Managed Policy Examples (p. 360)

The following shows an example of a permissions policy.

```
{
```

```
    "Version": "2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":["cloudwatch:GetMetricStatistics","cloudwatch:ListMetrics"],
        "Resource":"*",
        "Condition":{
            "Bool":{
                "aws:SecureTransport":"true"
                }
            }
        }
    ]
}
```

This sample policy has one statement that grants permissions to a group for two CloudWatch actions (`cloudwatch:GetMetricStatistics`data, and `cloudwatch:ListMetrics`), but only if the group uses SSL with the request (`"aws:SecureTransport":"true"`). For more information about the elements within an IAM policy statement, see Specifying Policy Elements: Actions, Effects, and Principals (p. 355) and IAM Policy Elements Reference in *IAM User Guide*.

Permissions Required to Use the CloudWatch Console

For a user to work with the CloudWatch console, that user must have a minimum set of permissions that allow the user to describe other AWS resources in their AWS account. The CloudWatch console requires permissions from the following services:

- Amazon EC2 Auto Scaling
- CloudTrail
- CloudWatch
- CloudWatch Events
- CloudWatch Logs
- Amazon EC2
- Amazon ES
- IAM
- Kinesis
- Lambda
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon SWF

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can still use the CloudWatch console, also attach the `CloudWatchReadOnlyAccess` managed policy to the user, as described in AWS Managed (Predefined) Policies for CloudWatch (p. 359).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the CloudWatch API.

The full set of permissions required to work with the CloudWatch console are listed below:

- application-autoscaling:DescribeScalingPolicies
- autoscaling:DescribeAutoScalingGroups
- autoscaling:DescribePolicies

- cloudtrail:DescribeTrails
- cloudwatch:DeleteAlarms
- cloudwatch:DescribeAlarmHistory
- cloudwatch:DescribeAlarms
- cloudwatch:GetMetricData
- cloudwatch:GetMetricStatistics
- cloudwatch:ListMetrics
- cloudwatch:PutMetricAlarm
- cloudwatch:PutMetricData
- ec2:DescribeInstances
- ec2:DescribeTags
- ec2:DescribeVolumes
- es:DescribeElasticsearchDomain
- es:ListDomainNames
- events:DeleteRule
- events:DescribeRule
- events:DisableRule
- events:EnableRule
- events:ListRules
- events:PutRule
- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStream
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction
- lambda:GetFunctionConfiguration
- lambda:ListAliases
- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup
- logs:CreateLogStream
- logs:DeleteLogGroup
- logs:DeleteLogStream
- logs:DeleteMetricFilter
- logs:DeleteRetentionPolicy

- logs:DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups
- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeSubscriptionFilters
- logs:FilterLogEvents
- logs:GetLogEvents
- logs:PutMetricFilter
- logs:PutRetentionPolicy
- logs:PutSubscriptionFilter
- logs:TestMetricFilter
- s3:CreateBucket
- s3:ListBucket
- sns:CreateTopic
- sns:GetTopicAttributes
- sns:ListSubscriptions
- sns:ListTopics
- sns:SetTopicAttributes
- sns:Subscribe
- sns:Unsubscribe
- sqs:GetQueueAttributes
- sqs:GetQueueUrl
- sqs:ListQueues
- sqs:SetQueueAttributes
- swf:CreateAction
- swf:DescribeAction
- swf:ListActionTemplates
- swf:RegisterAction
- swf:RegisterDomain
- swf:UpdateAction

AWS Managed (Predefined) Policies for CloudWatch

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so that you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to CloudWatch:

- **CloudWatchFullAccess** – Grants full access to CloudWatch.
- **CloudWatchReadOnlyAccess** – Grants read-only access to CloudWatch.
- **CloudWatchActionsEC2Access** – Grants read-only access to CloudWatch alarms and metrics in addition to Amazon EC2 metadata. Grants access to the Stop, Terminate, and Reboot API actions for EC2 instances.

Note
You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

You can also create your own custom IAM policies to allow permissions for CloudWatch actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various CloudWatch actions. These policies work when you are using the CloudWatch API, AWS SDKs, or the AWS CLI.

Examples

Example 1: Allow User Full Access to CloudWatch

The following policy allows a user to access all CloudWatch actions, CloudWatch Logs actions, Amazon SNS actions, and read-only access to Amazon EC2 Auto Scaling.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:*",
        "logs:*",
        "sns:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example 2: Allow Read-Only Access to CloudWatch

The following policy allows a user read-only access to CloudWatch and view Amazon EC2 Auto Scaling actions, CloudWatch metrics, CloudWatch Logs data, and alarm-related Amazon SNS data.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "logs:Get*",
        "logs:Describe*",
        "sns:Get*",
        "sns:List*"
```

```
        ],
        "Effect": "Allow",
        "Resource": "*"
    }
  ]
}
```

Example 3: Stop or Terminate an Amazon EC2 Instance

The following policy allows an CloudWatch alarm action to stop or terminate an EC2 instance. In the sample below, the GetMetricStatistics, ListMetrics, and DescribeAlarms actions are optional. It is recommended that you include these actions to ensure that you have correctly stopped or terminated the instance.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:ListMetrics",
        "cloudwatch:DescribeAlarms"
      ],
      "Sid": "00000000000000",
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Sid": "00000000000000",
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Using Service-Linked Roles for CloudWatch Alarms

Amazon CloudWatch uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to CloudWatch. Service-linked roles are predefined by CloudWatch and include all the permissions that the service requires to call other AWS services on your behalf.

The service-linked role in CloudWatch makes setting up CloudWatch alarms that can terminate, stop, or reboot Amazon EC2 instance easier because you don't have to manually add the necessary permissions. CloudWatch defines the permissions of the service-linked role, and unless defined otherwise, only CloudWatch can assume the role. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete the roles only after first deleting their related resources. This protects your CloudWatch resources because you can't inadvertently remove permissions to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-Linked Role Permissions for CloudWatch Alarms

CloudWatch uses the service-linked role named **AWSServiceRoleForCloudWatchEvents** – CloudWatch uses this service-linked role to perform Amazon EC2 alarm actions.

The AWSServiceRoleForCloudWatchEvents service-linked role trusts the CloudWatch Events service to assume the role. CloudWatch Events invokes the terminate, stop, or reboot instance actions when called upon by the alarm.

The AWSServiceRoleForCloudWatchEvents service-linked role permissions policy allows CloudWatch Events to complete the following actions on Amazon EC2 instances:

- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:RecoverInstances`
- `ec2:DescribeInstanceRecoveryAttribute`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceStatus`

Creating a Service-Linked Role for CloudWatch Alarms

You do not need to manually create a service-linked role. The first time you create an alarm in the AWS Management Console, the IAM CLI, or the IAM API, CloudWatch creates the service-linked role for you.

> **Important**
> This service-linked role can appear in your account if you completed an action in another service that uses the features supported by this role. Also, if you were using the CloudWatch service before January 1, 2017, when it began supporting service-linked roles, then CloudWatch created the AWSServiceRoleForCloudWatchEvents role in your account. To learn more, see A New Role Appeared in My IAM Account.

For more information, see Creating a Service-Linked Role in the *IAM User Guide*.

Creating a Service-Linked Role for CloudWatch Alarms

CloudWatch does not allow you to edit the **AWSServiceRoleForCloudWatchEvents** role. After you create the role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the **AWSServiceRoleForCloudWatchEvents** role using IAM.

Editing a Service-Linked Role Description (IAM Console)

You can use the IAM console to edit the description of a service-linked role.

To edit the description of a service-linked role (console)

1. In the navigation pane of the IAM console, choose **Roles**.

2. Choose the name of the role to modify.

3. To the far right of **Role description**, choose **Edit**.

4. Type a new description in the box and choose **Save**.

Editing a Service-Linked Role Description (AWS CLI)

You can use IAM commands from the AWS Command Line Interface to edit the description of a service-linked role.

To change the description of a service-linked role (AWS CLI)

1. (Optional) To view the current description for a role, use the following commands:

```
$ aws iam get-role --role-name role-name
```

Use the role name, not the ARN, to refer to roles with the AWS CLI commands. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, you refer to the role as **myrole**.

2. To update a service-linked role's description, use the following command:

```
$ aws iam update-role-description --role-name role-name --description description
```

Editing a Service-Linked Role Description (IAM API)

You can use the IAM API to edit the description of a service-linked role.

To change the description of a service-linked role (API)

1. (Optional) To view the current description for a role, use the following command:

   GetRole

2. To update a role's description, use the following command:

   UpdateRoleDescription

Deleting a Service-Linked Role for CloudWatch Alarms

If you no longer have alarms that automatically stop, terminate, or reboot EC2 instances, we recommend that you delete the AWSServiceRoleForCloudWatchEvents role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session in the IAM console

1. Open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane, choose **Roles**. Choose the name (not the check box) of the AWSServiceRoleForCloudWatchEvents role.

3. On the **Summary** page for the selected role, choose **Access Advisor** and review the recent activity for the service-linked role.

    > **Note**
    > If you are unsure whether CloudWatch is using the AWSServiceRoleForCloudWatchEvents role, try to delete the role. If the service is using the role, then the deletion fails and you can view the regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

Deleting a Service-Linked Role (IAM Console)

You can use the IAM console to delete a service-linked role.

To delete a service-linked role (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane, choose **Roles**. Select the check box next to AWSServiceRoleForCloudWatchEvents, not the name or row itself.

3. For **Role actions**, choose **Delete role**.

4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. To proceed, choose **Yes, Delete**.

5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, the deletion task can succeed or fail after you submit the role for deletion. If the task fails, choose **View details** or **View Resources** from the notifications to learn why the deletion failed. If the deletion fails because there are resources in the service that are being used by the role, then the reason for the failure includes a list of resources.

Deleting a Service-Linked Role (AWS CLI)

You can use IAM commands from the AWS Command Line Interface to delete a service-linked role.

To delete a service-linked role (AWS CLI)

1. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the `deletion-task-id` from the response to check the status of the deletion task. Type the following command to submit a service-linked role deletion request:

    ```
    $ aws iam delete-service-linked-role --role-name AWSServiceRoleForCloudWatchEvents
    ```

2. Type the following command to check the status of the deletion task:

    ```
    $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
    ```

    The status of the deletion task can be `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Deleting a Service-Linked Role (IAM API)

You can use the IAM API to delete a service-linked role.

To delete a service-linked role (API)

1. To submit a deletion request for a service-linked roll, call DeleteServiceLinkedRole. In the request, specify the AWSServiceRoleForCloudWatchEvents role name.

   Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the `DeletionTaskId` from the response to check the status of the deletion task.

2. To check the status of the deletion, call GetServiceLinkedRoleDeletionStatus. In the request, specify the `DeletionTaskId`.

   The status of the deletion task can be `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Amazon CloudWatch Permissions Reference

When you are setting up Access Control (p. 352) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The table lists each CloudWatch API operation and the corresponding actions for which you can grant permissions to perform the action. You specify the actions in the policy's `Action` field, and you specify a wildcard character (*) as the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your CloudWatch policies to express conditions. For a complete list of AWS-wide keys, see AWS Global and IAM Condition Context Keys in the *IAM User Guide*.

> **Note**
> To specify an action, use the `cloudwatch:` prefix followed by the API operation name.
> For example: `cloudwatch:GetMetricStatistics`, `cloudwatch:ListMetrics`, or
> `cloudwatch:*` (for all CloudWatch actions).

Tables

- CloudWatch API Operations and Required Permissions
- CloudWatch Events API Operations and Required Permissions
- CloudWatch Logs API Operations and Required Permissions
- Amazon EC2 API Operations and Required Permissions
- Amazon EC2 Auto Scaling API Operations and Required Permissions

CloudWatch API Operations and Required Permissions for Actions

CloudWatch API Operations	Required Permissions (API Actions)
DeleteAlarms	cloudwatch:DeleteAlarms Required to delete an alarm.
DeleteDashboards	cloudwatch:DeleteDashboards Required to delete a dashboard.
DescribeAlarmHistory	cloudwatch:DescribeAlarmHistory

CloudWatch API Operations	Required Permissions (API Actions)
	Required to view alarm history.
DescribeAlarms	`cloudwatch:DescribeAlarms` Required to retrieve alarm information by name.
DescribeAlarmsForMetric	`cloudwatch:DescribeAlarmsForMetric` Required to view alarms for a metric.
DisableAlarmActions	`cloudwatch:DisableAlarmActions` Required to disable an alarm action.
EnableAlarmActions	`cloudwatch:EnableAlarmActions` Required to enable an alarm action.
GetDashboard	`cloudwatch:GetDashboard` Required to display data about existing dashboards.
GetMetricData	`cloudwatch:GetMetricData` Required to retrieve large batches of metric data and perform metric math on that data.
GetMetricStatistics	`cloudwatch:GetMetricStatistics` Required to view graphs in other parts of the CloudWatch console and in dashboard widgets.
ListDashboards	`cloudwatch:ListDashboards` Required to view the list of CloudWatch dashboards in your account.
ListMetrics	`cloudwatch:ListMetrics` Required to view or search metric names within the CloudWatch console and in the CLI. Required to select metrics on dashboard widgets.
PutDashboard	`cloudwatch:PutDashboard` Required to create a dashboard or update an existing dashboard.
PutMetricAlarm	`cloudwatch:PutMetricAlarm` Required to create or update an alarm.
PutMetricData	`cloudwatch:PutMetricData` Required to create metrics.
SetAlarmState	`cloudwatch:SetAlarmState` Required to manually set an alarm's state.

CloudWatch Events API Operations and Required Permissions for Actions

CloudWatch Events API Operations	Required Permissions (API Actions)
DeleteRule	`events:DeleteRule` Required to delete a rule.
DescribeRule	`events:DescribeRule` Required to list the details about a rule.
DisableRule	`events:DisableRule` Required to disable a rule.
EnableRule	`events:EnableRule` Required to enable a rule.
ListRuleNamesByTarget	`events:ListRuleNamesByTarget` Required to list rules associated with a target.
ListRules	`events:ListRules` Required to list all rules in your account.
ListTargetsByRule	`events:ListTargetsByRule` Required to list all targets associated with a rule.
PutEvents	`events:PutEvents` Required to add custom events that can be matched to rules.
PutRule	`events:PutRule` Required to create or update a rule.
PutTargets	`events:PutTargets` Required to add targets to a rule.
RemoveTargets	`events:RemoveTargets` Required to remove a target from a rule.
TestEventPattern	`events:TestEventPattern` Required to test an event pattern against a given event.

CloudWatch Logs API Operations and Required Permissions for Actions

CloudWatch Logs API Operations	Required Permissions (API Actions)
CancelExportTask	`logs:CancelExportTask`

CloudWatch Logs API Operations	Required Permissions (API Actions)
	Required to cancel a pending or running export task.
CreateExportTask	`logs:CreateExportTask` Required to export data from a log group to an Amazon S3 bucket.
CreateLogGroup	`logs:CreateLogGroup` Required to create a new log group.
CreateLogStream	`logs:CreateLogStream` Required to create a new log stream in a log group.
DeleteDestination	`logs:DeleteDestination` Required to delete a log destination and disables any subscription filters to it.
DeleteLogGroup	`logs:DeleteLogGroup` Required to delete a log group and any associated archived log events.
DeleteLogStream	`logs:DeleteLogStream` Required to delete a log stream and any associated archived log events.
DeleteMetricFilter	`logs:DeleteMetricFilter` Required to delete a metric filter associated with a log group.
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` Required to delete a log group's retention policy.
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` Required to delete the subscription filter associated with a log group.
DescribeDestinations	`logs:DescribeDestinations` Required to view all destinations associated with the account.
DescribeExportTasks	`logs:DescribeExportTasks` Required to view all export tasks associated with the account.

CloudWatch Logs API Operations	Required Permissions (API Actions)
DescribeLogGroups	`logs:DescribeLogGroups` Required to view all log groups associated with the account.
DescribeLogStreams	`logs:DescribeLogStreams` Required to view all log streams associated with a log group.
DescribeMetricFilters	`logs:DescribeMetricFilters` Required to view all metrics associated with a log group.
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` Required to view all subscription filters associated with a log group.
FilterLogEvents	`logs:FilterLogEvents` Required to sort log events by log group filter pattern.
GetLogEvents	`logs:GetLogEvents` Required to retrieve log events from a log stream.
ListTagsLogGroup	`logs:ListTagsLogGroup` Required to list the tags associated with a log group.
PutDestination	`logs:PutDestination` Required to create or update a destination log stream (such as a Kinesis stream).
PutDestinationPolicy	`logs:PutDestinationPolicy` Required to create or update an access policy associated with an existing log destination.
PutLogEvents	`logs:PutLogEvents` Required to upload a batch of log events to a log stream.
PutMetricFilter	`logs:PutMetricFilter` Required to create or update a metric filter and associate it with a log group.
PutRetentionPolicy	`logs:PutRetentionPolicy` Required to set the number of days to keep log events (retention) in a log group.

CloudWatch Logs API Operations	Required Permissions (API Actions)
PutSubscriptionFilter	`logs:PutSubscriptionFilter` Required to create or update a subscription filter and associate it with a log group.
TestMetricFilter	`logs:TestMetricFilter` Required to test a filter pattern against a sampling of log event messages.

Amazon EC2 API Operations and Required Permissions for Actions

Amazon EC2 API Operations	Required Permissions (API Actions)
DescribeInstanceStatus	`ec2:DescribeInstanceStatus` Required to view EC2 instance status details.
DescribeInstances	`ec2:DescribeInstances` Required to view EC2 instance details.
RebootInstances	`ec2:RebootInstances` Required to reboot an EC2 instance.
StopInstances	`ec2:StopInstances` Required to stop an EC2 instance.
TerminateInstances	`ec2:TerminateInstances` Required to terminate an EC2 instance.

Amazon EC2 Auto Scaling API Operations and Required Permissions for Actions

Amazon EC2 Auto Scaling API Operations	Required Permissions (API Actions)
Scaling	`autoscaling:Scaling` Required to scale an Auto Scaling group.
Trigger	`autoscaling:Trigger` Required to trigger an Auto Scaling action.

Logging Amazon CloudWatch API Calls with AWS CloudTrail

Amazon CloudWatch is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CloudWatch. CloudTrail captures API calls made by or on behalf of your AWS account. The calls captured include calls from the CloudWatch console and code calls to the CloudWatch API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for CloudWatch. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to CloudWatch, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

Topics
- CloudWatch Information in CloudTrail (p. 371)
- Example: CloudWatch Log File Entries (p. 372)

CloudWatch Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in CloudWatch, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for CloudWatch, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

CloudWatch supports logging the following actions as events in CloudTrail log files:

- DeleteAlarms
- DeleteDashboards
- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DisableAlarmActions

- EnableAlarmActions
- GetDashboard
- ListDashboards
- PutDashboard
- PutMetricAlarm
- SetAlarmState

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Example: CloudWatch Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the **PutMetricAlarm** action.

```
{
    "Records": [{
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "Root",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID"
        },
        "eventTime": "2014-03-23T21:50:34Z",
        "eventSource": "monitoring.amazonaws.com",
        "eventName": "PutMetricAlarm",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "threshold": 50.0,
            "period": 60,
            "metricName": "CloudTrail Test",
            "evaluationPeriods": 3,
            "comparisonOperator": "GreaterThanThreshold",
            "namespace": "AWS/CloudWatch",
            "alarmName": "CloudTrail Test Alarm",
            "statistic": "Sum"
        },
        "responseElements": null,
        "requestID": "29184022-b2d5-11e3-a63d-9b463e6d0ff0",
        "eventID": "b096d5b7-dcf2-4399-998b-5a53eca76a27"
```

```
    },
    ..additional entries
  ]
  }
```

The following log file entry shows that a user called the CloudWatch Events **PutRule** action.

```
{
        "eventVersion":"1.03",
        "userIdentity":{
            "type":"Root",
            "principalId":"123456789012",
            "arn":"arn:aws:iam::123456789012:root",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext":{
                "attributes":{
                    "mfaAuthenticated":"false",
                    "creationDate":"2015-11-17T23:56:15Z"
                }
            }
        },
        "eventTime":"2015-11-18T00:11:28Z",
        "eventSource":"events.amazonaws.com",
        "eventName":"PutRule",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"AWS Internal",
        "userAgent":"AWS CloudWatch Console",
        "requestParameters":{
            "description":"",
            "name":"cttest2",
            "state":"ENABLED",
            "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance
 State-change Notification\"]}",
            "scheduleExpression":""
        },
        "responseElements":{
            "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
        },
        "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
        "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
        "eventType":"AwsApiCall",
        "apiVersion":"2015-10-07",
        "recipientAccountId":"123456789012"
}
```

The following log file entry shows that a user called the CloudWatch Logs **CreateExportTask** action.

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
```

```
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```

Document History

The following table describes important changes in each release of the CloudWatch User Guide, beginning in June 2018. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
Enhancements to the CloudWatch Agent (p. 375)	The CloudWatch agent has been updated to work with both the StatsD and collectd protocols. It also has improved cross-account support. For more information, see Retrieve Custom Metrics with StatsD, Retrieve Custom Metrics with collectd, and Sending Metrics and Logs to a Different AWS Account in the *Amazon CloudWatch User Guide*.	September 28, 2018
Support for Amazon VPC endpoints (p. 375)	You can now establish a private connection between your VPC and CloudWatch. For more information, see Using CloudWatch with Interface VPC Endpoints in the *Amazon CloudWatch User Guide*.	June 28, 2018

The following table describes important changes to the Amazon CloudWatch User Guide before June 2018.

Change	Description	Release Date
Metric math	You can now perform math expressions on CloudWatch metrics, producing new time series that you can add to graphs on your dashboard. For more information, see Use Metric Math (p. 47).	4 April 2018
"M out of N" alarms	You can now configure an alarm to trigger based on "M out of N" datapoints in any alarm evaluation interval. For more information, see Evaluating an Alarm (p. 327).	8 December 2017
CloudWatch agent	A new unified CloudWatch agent was released. You can use the unified multi-platform agent to collect custom both system metrics and log files from Amazon EC2 instances and on-premises servers. The new agent supports both Windows and Linux and enables customization of metrics collected, including sub-resource metrics such as per-CPU core. For more information, see Collect Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent (p. 53).	7 September 2017

Change	Description	Release Date
NAT gateway metrics	Added metrics for Amazon VPC NAT gateway. For more information, see Amazon VPC NAT Gateway Metrics and Dimensions (p. 318).	7 September 2017
High-resolution metrics	You can now optionally set up custom metrics as high-resolution metrics, with a granularity of as low as one second. For more information, see High-Resolution Metrics (p. 45).	26 July 2017
Dashboard APIs	You can now create, modify, and delete dashboards using APIs and the AWS CLI. For more information, see Create a CloudWatch Dashboard (p. 19).	6 July 2017
AWS Direct Connect metrics	Added metrics for AWS Direct Connect. For more information, see AWS Direct Connect Metrics and Dimensions (p. 137).	29 June 2017
Amazon VPC VPN metrics	Added metrics for Amazon VPC VPN. For more information, see Amazon VPC VPN Metrics and Dimensions (p. 322).	15 May 2017
AppStream 2.0 metrics	Added metrics for AppStream 2.0. For more information, see AppStream 2.0 Metrics and Dimensions (p. 120).	8 March 2017
CloudWatch console color picker	You can now choose the color for each metric on your dashboard widgets. For more information, see Edit a Graph on a CloudWatch Dashboard (p. 22).	27 February 2017
Alarms on dashboards	Alarms can now be added to dashboards. For more information, see Add or Remove an Alarm from a CloudWatch Dashboard (p. 26).	15 February 2017
Added metrics for Amazon Polly	Added metrics for Amazon Polly. For more information, see Amazon Polly Metrics (p. 272).	1 December 2016
Added metrics for Amazon Kinesis Data Analytics	Added metrics for Amazon Kinesis Data Analytics. For more information, see Amazon Kinesis Data Analytics Metrics (p. 237).	1 December 2016
Added support for percentile statistics	You can specify any percentile, using up to two decimal places (for example, p95.45). For more information, see Percentiles (p. 7).	17 November 2016
Added metrics for Amazon Simple Email Service	Added metrics for Amazon Simple Email Service. For more information, see Amazon Simple Email Service Metrics and Dimensions (p. 290).	2 November 2016
Updated metrics retention	Amazon CloudWatch now retains metrics data for 15 months instead of 14 days.	1 November 2016
Updated metrics console interface	The CloudWatch console is updated with improvements to existing functionality and new functionality.	1 November 2016
Added metrics for Amazon Elastic Transcoder	Added metrics for Amazon Elastic Transcoder. For more information, see Amazon Elastic Transcoder Metrics and Dimensions (p. 206).	20 September 2016

Change	Description	Release Date
Added metrics for Amazon API Gateway	Added metrics for Amazon API Gateway. For more information, see Amazon API Gateway Metrics and Dimensions (p. 118).	9 September 2016
Added metrics for AWS Key Management Service	Added metrics for AWS Key Management Service. For more information, see AWS Key Management Service Metrics and Dimensions (p. 255).	9 September 2016
Added metrics for the new Application Load Balancers supported by Elastic Load Balancing	Added metrics for Application Load Balancers. For more information, see Elastic Load Balancing Metrics and Dimensions (p. 178).	11 August 2016
Added new NetworkPacketsIn and NetworkPacketsOut metrics for Amazon EC2	Added new NetworkPacketsIn and NetworkPacketsOut metrics for Amazon EC2. For more information, see Amazon EC2 Metrics and Dimensions (p. 152).	23 March 2016
Added new metrics for Amazon EC2 Spot fleet	Added new metrics for Amazon EC2 Spot fleet. For more information, see Amazon EC2 Spot Fleet Metrics and Dimensions (p. 159).	21 March 2016
Added new CloudWatch Logs metrics	Added new CloudWatch Logs metrics. For more information, see Amazon CloudWatch Logs Metrics and Dimensions (p. 134).	10 March 2016
Added Amazon Elasticsearch Service and AWS WAF metrics and dimensions	Added Amazon Elasticsearch Service and AWS WAF metrics and dimensions. For more information, see Amazon Elasticsearch Service Metrics and Dimensions (p. 201) and AWS WAF Metrics and Dimensions (p. 323).	14 October 2015
Added support for CloudWatch dashboards	Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those that are spread out across different regions. For more information, see Using Amazon CloudWatch Dashboards (p. 19).	8 October 2015
Added AWS Lambda metrics and dimensions	Added AWS Lambda metrics and dimensions. For more information, see AWS Lambda Metrics and Dimensions (p. 256).	4 September 2015
Added Amazon Elastic Container Service metrics and dimensions	Added Amazon Elastic Container Service metrics and dimensions. For more information, see Amazon ECS Metrics and Dimensions (p. 161).	17 August 2015

Change	Description	Release Date
Added Amazon Simple Storage Service metrics and dimensions	Added Amazon Simple Storage Service metrics and dimensions. For more information, see Amazon Simple Storage Service Metrics and Dimensions (p. 297).	26 July 2015
New feature: Reboot alarm action	Added the reboot alarm action and new IAM role for use with alarm actions. For more information, see Create Alarms to Stop, Terminate, Reboot, or Recover an Instance (p. 340).	23 July 2015
Added Amazon WorkSpaces metrics and dimensions	Added Amazon WorkSpaces metrics and dimensions. For more information, see Amazon WorkSpaces Metrics and Dimensions (p. 324).	30 April 2015
Added Amazon Machine Learning metrics and dimensions	Added Amazon Machine Learning metrics and dimensions. For more information, see Amazon Machine Learning Metrics and Dimensions (p. 262).	9 April 2015
New feature: Amazon EC2 instance recovery alarm actions	Updated alarm actions to include new EC2 instance recovery action. For more information, see Create Alarms to Stop, Terminate, Reboot, or Recover an Instance (p. 340).	12 March 2015
Added Amazon CloudFront and Amazon CloudSearch metrics and dimensions	Added Amazon CloudFront and Amazon CloudSearch metrics and dimensions. For more information, see Amazon CloudFront Metrics and Dimensions (p. 123) and Amazon CloudSearch Metrics and Dimensions (p. 124).	6 March 2015
Added Amazon Simple Workflow Service metrics and dimensions	Added Amazon Simple Workflow Service metrics and dimensions. For more information, see Amazon SWF Metrics and Dimensions (p. 303).	9 May 2014
Updated guide to add support for AWS CloudTrail	Added a new topic to explain how you can use AWS CloudTrail to log activity in Amazon CloudWatch. For more information, see Logging Amazon CloudWatch API Calls with AWS CloudTrail (p. 371).	30 April 2014
Updated guide to use the new AWS Command Line Interface (AWS CLI)	The AWS CLI is a cross-service CLI with a simplified installation, unified configuration, and consistent command line syntax. The AWS CLI is supported on Linux/Unix, Windows, and Mac. The CLI examples in this guide have been updated to use the new AWS CLI. For information about how to install and configure the new AWS CLI, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.	21 February 2014

Change	Description	Release Date
Added Amazon Redshift and AWS OpsWorks metrics and dimensions	Added Amazon Redshift and AWS OpsWorks metrics and dimensions. For more information, see Amazon Redshift Metrics and Dimensions (p. 273) and AWS OpsWorks Metrics and Dimensions (p. 268).	16 July 2013
Added Amazon Route 53 metrics and dimensions	Added Amazon Route 53 metrics and dimensions. For more information, see Route 53 Metrics and Dimensions (p. 285).	26 June 2013
New feature: Amazon CloudWatch Alarm Actions	Added a new section to document Amazon CloudWatch alarm actions, which you can use to stop or terminate an Amazon Elastic Compute Cloud instance. For more information, see Create Alarms to Stop, Terminate, Reboot, or Recover an Instance (p. 340).	8 January 2013
Updated EBS metrics	Updated the EBS metrics to include two new metrics for Provisioned IOPS volumes. For more information, see Amazon EBS Metrics and Dimensions (p. 172).	20 November 2012
New billing alerts	You can now monitor your AWS charges using Amazon CloudWatch metrics and create alarms to notify you when you have exceeded the specified threshold. For more information, see Create a Billing Alarm to Monitor Your Estimated AWS Charges (p. 346).	10 May 2012
New metrics	You can now access six new Elastic Load Balancing metrics that provide counts of various HTTP response codes. For more information, see Elastic Load Balancing Metrics and Dimensions (p. 178).	19 October 2011
New feature	You can now access metrics from Amazon EMR. For more information, see Amazon EMR Metrics and Dimensions (p. 191).	30 June 2011
New feature	You can now access metrics from Amazon Simple Notification Service and Amazon Simple Queue Service. For more information, see Amazon Simple Notification Service Metrics and Dimensions (p. 292) and Amazon SQS Metrics and Dimensions (p. 295).	14 July 2011
New Feature	Added information about using the `PutMetricData` API to publish custom metrics. For more information, see Publish Custom Metrics (p. 45).	10 May 2011
Updated metrics retention	Amazon CloudWatch now retains the history of an alarm for two weeks rather than six weeks. With this change, the retention period for alarms matches the retention period for metrics data.	07 April 2011
New feature	Added ability to send Amazon Simple Notification Service or Auto Scaling notifications when a metric has crossed a threshold. For more information, see Alarms (p. 7).	02 December 2010

Change	Description	Release Date
New feature	A number of CloudWatch actions now include the MaxRecords and NextToken parameters, which enable you to control pages of results to display.	02 December 2010
New feature	This service now integrates with AWS Identity and Access Management (IAM).	02 December 2010