Amazon CloudWatch

User Guide

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.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications, and display custom collections of metrics that you choose.

You can create alarms which watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. 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.

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.

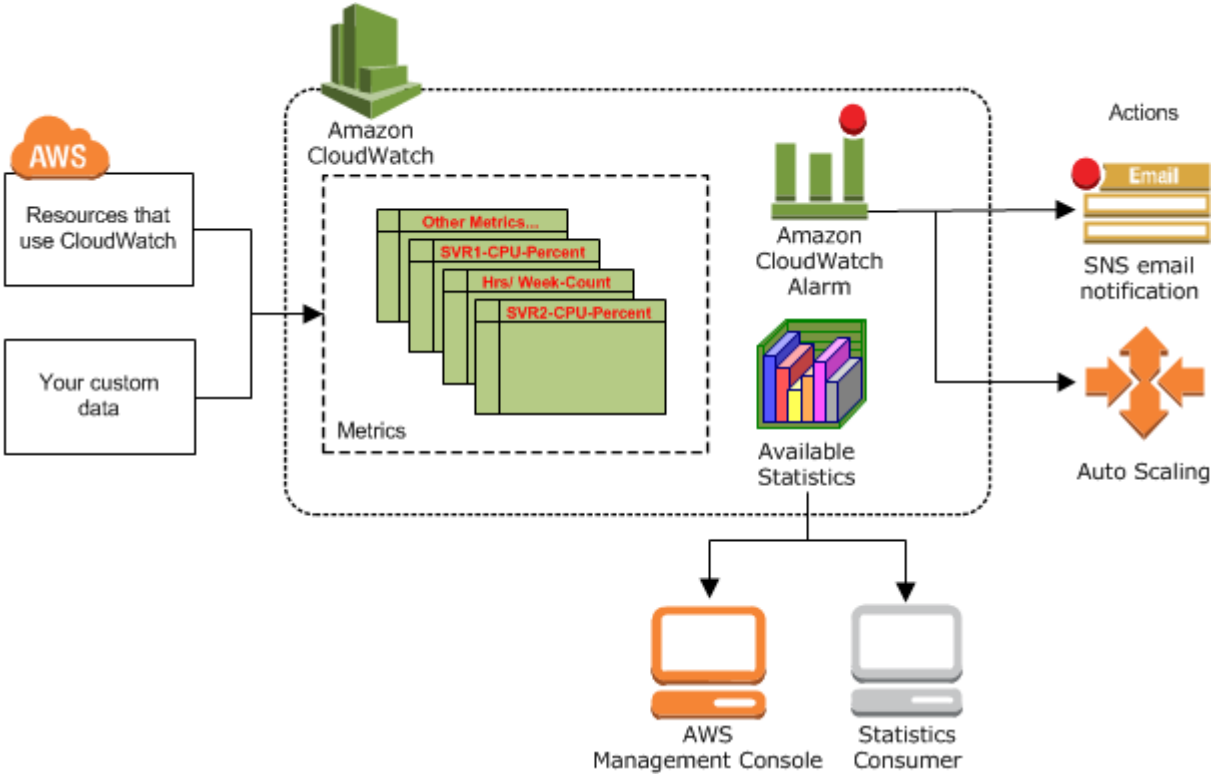
Related AWS Services

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. 65).
- **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. 200).
- **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. 180).

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 AWS Services That Publish CloudWatch Metrics (p. 162).

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)

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 Services That Publish CloudWatch Metrics (p. 162).

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 as representing 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 in a metric has a time stamp, and (optionally) a unit of measure. You can retrieve statistics from CloudWatch for any metric.

For more information, see Viewing Available Metrics (p. 28) and Publishing Custom Metrics (p. 44).

Time Stamps

Each metric data point must be associated 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 is part of the identity of 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. 44).

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 Publishing Custom Metrics (p. 44).

For more information, see Using Amazon CloudWatch Alarms (p. 60) and Creating an Alarm from a Metric on a Graph (p. 43).

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.
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.

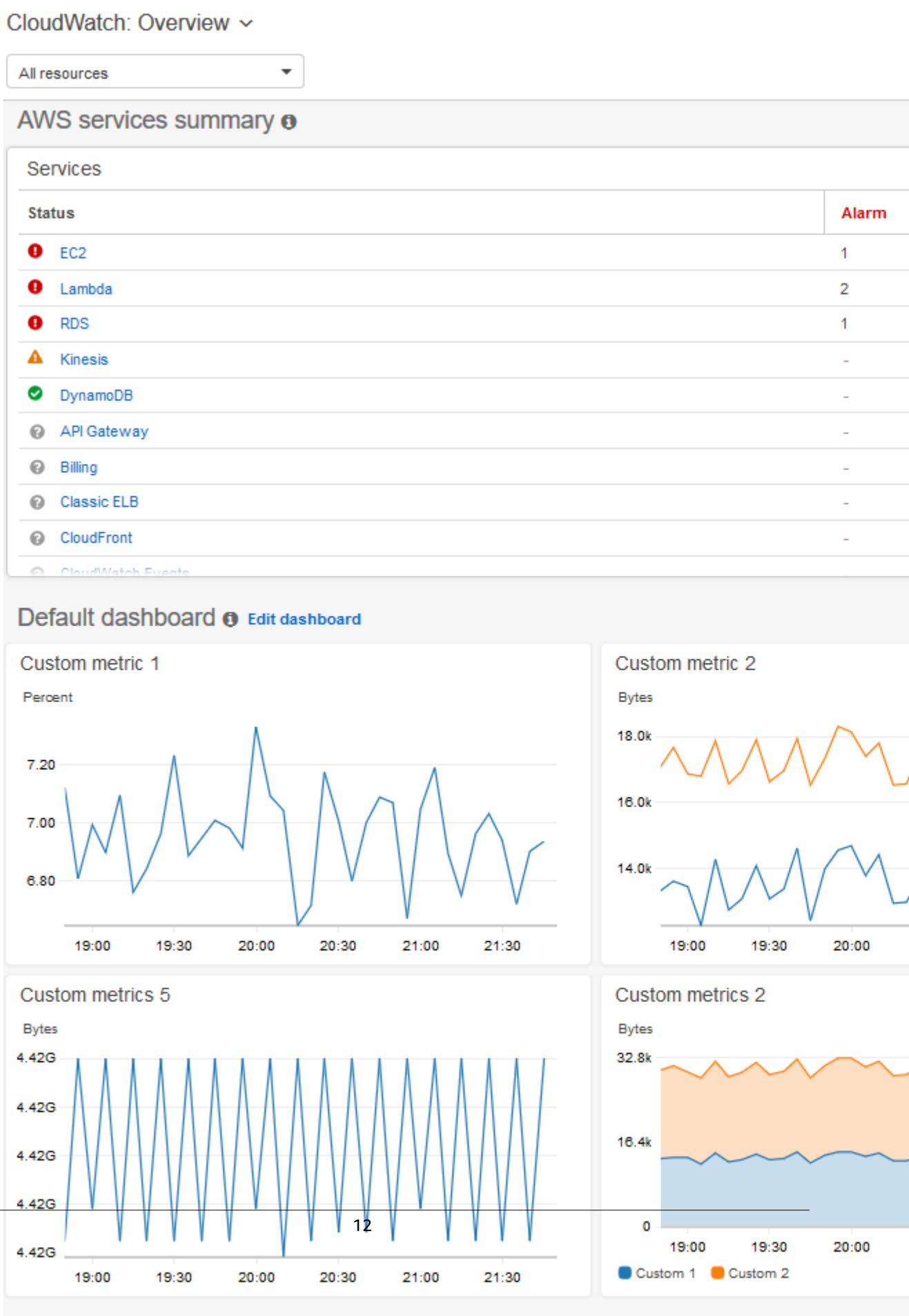
For information about how to install and configure the AWS CLI, see Getting Set Up with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

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

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

The CloudWatch overview home page appears.


CloudWatch: Overview
All resources
AWS services summary
Services
Status
Alarm
EC2
1
Lambda
2
RDS
1
Kinesis
-
DynamoDB
-
API Gateway
-
Billing
-
Classic ELB
-
CloudFront
-
Default dashboard
Edit dashboard
Custom metric 1
Percent
7.20
7.00
6.80
19:00
19:30
20:00
20:30
21:00
21:30
Custom metric 2
Bytes
18.0k
16.0k
14.0k
19:00
19:30
20:00
Custom metrics 5
Bytes
4.42G
4.42G
4.42G
4.42G
4.42G
19:00
19:30
20:00
20:30
21:00
21:30
Custom metrics 2
Bytes
32.8k
16.4k
0
19:00
19:30
20:00
Custom 1
Custom 2

The overview displays the following items, refreshed automatically.

- The upper left shows a list of AWS services you use in your account, along with the state of alarms in those services. The upper right shows two or four alarms in your account, depending on how many AWS services you use. The alarms shown are those in the ALARM state or those that most recently changed state.

  These upper areas enable you to assess the health of your AWS services, by seeing the alarm states in every service and the alarms that most recently changed state. This helps you monitor and quickly diagnose issues.
- Below these areas is the custom dashboard that you have created and named **CloudWatch-Default**, if any. This is a convenient way for you to add metrics about your own custom services or applications to the overview page, or to bring forward additional key metrics from AWS services that you most want to monitor.
- If you use six or more AWS services, below the default dashboard is a link to the automatic cross-service dashboard. The cross-service dashboard automatically displays key metrics from every AWS service you use, without requiring you to choose what metrics to monitor or create custom dashboards. You can also use it to drill down to any AWS service and see even more key metrics for that service.

  If you use fewer than six AWS services, the cross-service dashboard is shown automatically on this page.

From this overview, you can focus your view to a specific resource group or a specific AWS service. This enables you to narrow your view to a subset of resources in which you are interested. Using resource groups enables you to use tags to organize projects, focus on a subset of your architecture, or just distinguish between your production and development environments. For more information, see What Is AWS Resource Groups?.

Topics

See Key Metrics From All AWS Services

If you use six or more AWS services, the cross-service dashboard is not displayed on the overview page. You can switch to this dashboard to see key metrics from all the AWS services that you are using.

To open the cross service dashboard

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

   The overview appears.
2. Near the bottom of the page, choose **View cross service dashboard**.

   The cross-service dashboard appears, showing each AWS service you are using, displayed in alphabetical order. For each service, one or two key metrics are displayed.
3. You can focus on a particular service in two ways:

   a. To see more key metrics for a service, choose its name from the list at the top of the screen, where **Cross service dashboard** is currently shown. Or, you can choose **View *Service* dashboard** next to the service name.

An automatic dashboard for that service is displayed, showing more metrics for that service. Additionally, for some services, the bottom of the service dashboard displays resources related to that service. You can choose one of those resources to that service console and focus further on that resource.

b. To see all the alarms related to a service, choose the button on the right of the screen next to that service name. The text on this button indicates how many alarms you have created in this service, and whether any are in the ALARM state.

When the alarms are displayed, multiple alarms that have similar settings (such as dimensions, threshold, or period) may be shown in a single graph.

You can then view details about an alarm and see the alarm history. To do so, hover on the alarm graph, and choose the actions icon, **View in alarms**.

The alarms view appears in a new browser tab, displaying a list of your alarms, along with details about the chosen alarm. To see the history for this alarm, choose the **History** tab.

4. You can focus on resources in a particular resource group. To do so, choose the resource group from the list at the top of the page where **All resources** is displayed.

For more information, see Focus on Metrics and Alarms in a Resource Group (p. 16).

5. To change the time range shown in all graphs and alarms currently displayed, select the range you want next to **Time range** at the top of the screen. Choose **custom** to select from more time range options than those displayed by default.
6. Alarms are always refreshed once a minute. To refresh the view, choose the refresh icon (two curved arrows) at the top right of the screen. To change the automatic refresh rate for items on the screen other than alarms, choose the down arrow next to the refresh icon and choose the refresh rate you want. You can also choose to turn off automatic refresh.

Remove a Service from Appearing in the Cross Service Dashboard

You can prevent a service's metrics from appearing in the cross service dashboard. This helps you focus your cross service dashboard on the services you most want to monitor.

If you remove a service from the cross service dashboard, the alarms for that service still appear in the views of your alarms.

To remove a service's metrics from the cross service dashboard

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

The home page appears.

2. At the top of the page, under **Overview**, choose the service you want to remove.

The view changes to show metrics from only that service.

3. Choose **Actions**, then clear the check box next to **Show on cross service dashboard**.

Focus on Metrics and Alarms in a Single AWS Service

On the CloudWatch home page, you can focus the view to a single AWS service. You can drill down further by focusing on both a single AWS service and a resource group at the same time. The following procedure shows only how to focus on an AWS service.

To focus on a single service

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

   The home page appears.
2. Choose the service name from the list at the top of the screen, where **Overview** is currently shown.

   The view changes to display graphs of key metrics from the selected service.
3. To switch to viewing the alarms for this service, choose **Alarms dashboard** at the top of the screen where **Service dashboard** is currently displayed.
4. When viewing metrics, you can focus on a particular metric in several ways:

   a. To see more details about the metrics in any graph, hover on the graph, and choose the actions icon, **View in metrics**.

      The graph appears in a new tab, with the relevant metrics listed below the graph. You can customize your view of this graph, changing the metrics and resources shown, the statistic, the period, and other factors to get a better understanding of the current situation.

   b. You can view log events from the time range shown in the graph. This may help you discover events that happened in your infrastructure that are causing an unexpected change in your metrics.

      To see the log events, hover on the graph, and choose the actions icon, **View in logs**.

      The CloudWatch Logs view appears in a new tab, displaying a list of your log groups. To see the log events in one of these log groups that occurred during the time range shown in the original graph, choose that log group.
5. When viewing alarms, you can focus on a particular alarm in several ways:

   - To see more details about an alarm, hover on the alarm, and choose the actions icon, **View in alarms**.

     The alarms view appears in a new tab, displaying a list of your alarms, along with details about the chosen alarm. To see the history for this alarm, choose the **History** tab.
6. Alarms are always refreshed one time per minute. To refresh the view, choose the refresh icon (two curved arrows) at the top right of the screen. To change the automatic refresh rate for items on the screen other than alarms, choose the down arrow next to the refresh icon and choose a refresh ratet. You can also choose to turn off automatic refresh.
7. To change the time range shown in all graphs and alarms currently displayed, next to **Time range** at the top of the screen, choose the range . To select from more time range options than those displayed by default, choose **custom**.
8. To return to the cross-service dashboard, choose **Overview** in the list at the top of the screen that currently shows the service you are focusing on.

   Alternatively, from any view, you can choose **CloudWatch** at the top of the screen to clear all filters and return to the overview page.

Focus on Metrics and Alarms in a Resource Group

You can focus your view to display metrics and alarms from a single resource group. Using resource groups enables you to use tags to organize projects, focus on a subset of your architecture, or distinguish between your production and development environments. They also enable you to focus on each of these resource groups on the CloudWatch overview. For more information, see What Is AWS Resource Groups?.

When you focus on a resource group, the display changes to show only the services where you have tagged resources as part of this resource group. The recent alarms area shows only alarms related to the resource group. Additionally, if you have created a dashboard with the name **CloudWatch-Default-*ResourceGroupName***, it is displayed in the **Default dashboard** area.

You can drill down further by focusing on both a single AWS service and a resource group at the same time. The following procedure shows just how to focus on a resource group.

To focus on a single resource group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. At the top of the page, where **All resources** is displayed, choose a resource group.
3. To see more metrics related to this resource group, near the bottom of the screen, choose **View cross service dashboard**.

   The cross-service dashboard appears, showing only the services related to this resource group. For each service, one or two key metrics are displayed.
4. To change the time range shown in all graphs and alarms currently displayed, for **Time range** at the top of the screen, select a range. To select from more time range options than those displayed by default, choose **custom**.
5. Alarms are always refreshed one time per minute. To refresh the view, choose the refresh icon (two curved arrows) at the top right of the screen. To change the automatic refresh rate for items on the screen other than alarms, choose the down arrow next to the refresh icon and choose a refresh rate. You can also choose to turn off automatic refresh.

   Alarms are always refreshed one time per minute.
6. To return to showing information about all the resources in your account, near the top of the screen where the name of the resource group is currently displayed, choose **All resources**.

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**.

   If you use the name **CloudWatch-Default**, the dashboard appears on the overview on the CloudWatch home page. For more information, see Getting Started with Amazon CloudWatch (p. 11).

   If you use resource groups and name the dashboard **CloudWatch-Default-*ResourceGroupName***, it appears on the CloudWatch home page when you focus on that resource group.
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. 22).
   - 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. 22).
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 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. 18).
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 Period Override Setting or Refresh Interval for the CloudWatch Dashboard

You can specify how the period setting of graphs added to this dashboard are retained or modified.

To change the period override options

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose **Actions**.
3. Under **Period override**, choose one of the following:

   - Choose **Auto** to have the period of the metrics on each graph automatically adapt to the dashboard's time range.
   - Choose **Do not override** to ensure that the period setting of each graph is always obeyed.

- Choose one of the other options to cause graphs added to the dashboard to always adapt that chosen time as their period setting.

The **Period override** always reverts to **Auto** when the dashboard is closed or the browser is refreshed. Different settings for **Period override** cannot be saved.

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. 44).

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

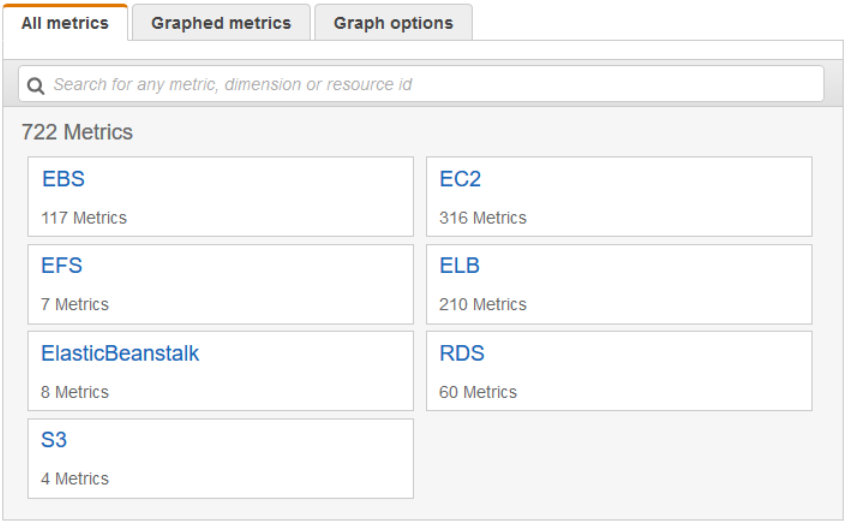
Viewing Available Metrics

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.

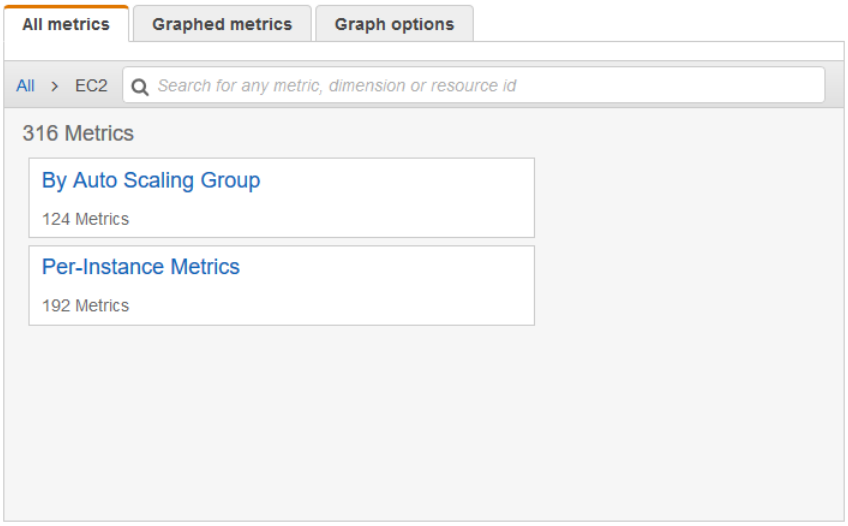
Only the AWS services that you're using send metrics to Amazon CloudWatch.

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**).



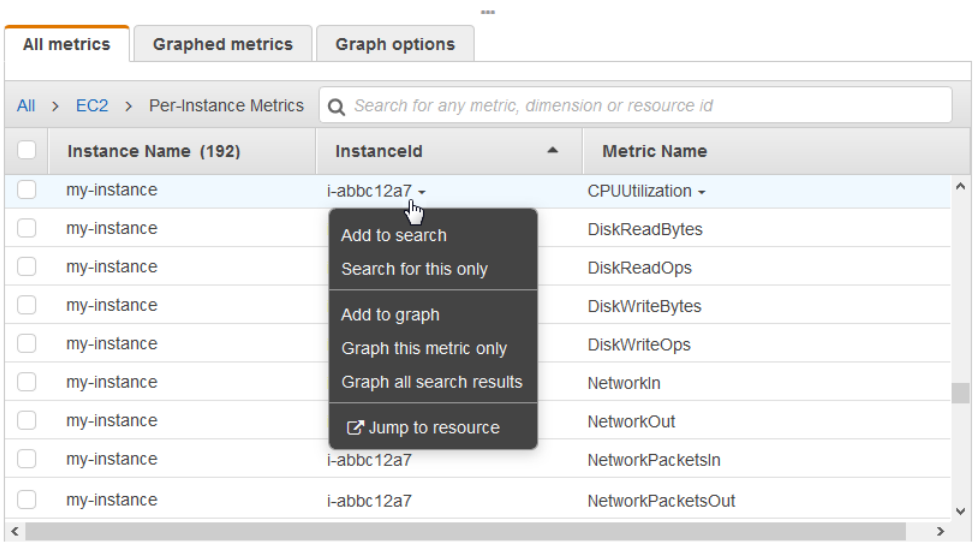
5. The **All metrics** tab displays all metrics for that dimension in the namespace. You can do the following:

   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 the namespaces, metrics, and dimensions for all services that publish metrics, see AWS Services That Publish CloudWatch Metrics (p. 162).

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
```

Searching for Available Metrics

You can search within all of 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, enter a search term, such as a metric name, namespace, dimension name or value, or resource name. This shows you all of 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.

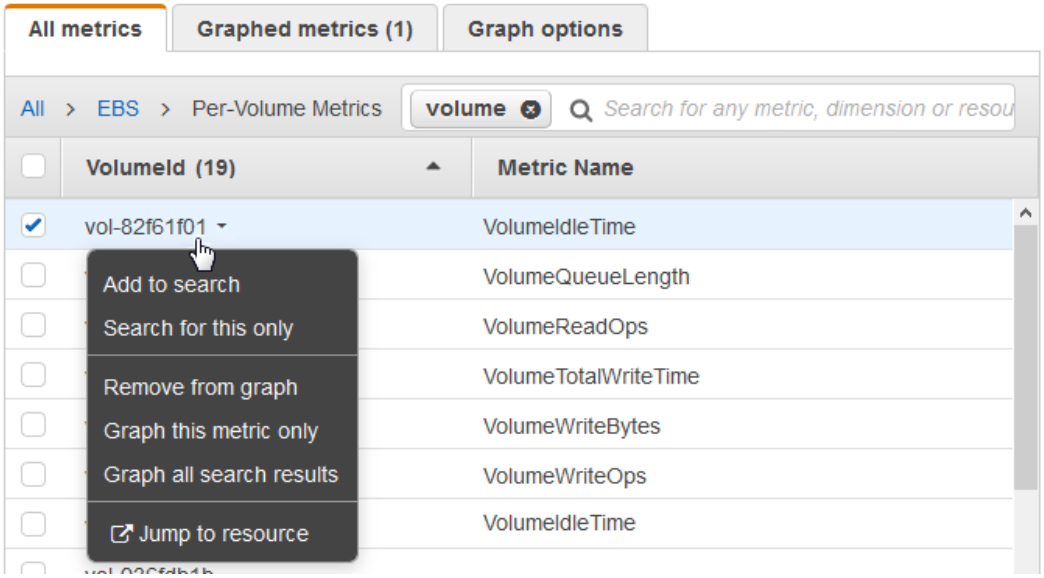
   For more information on search, see Using Search Expressions in Graphs (p. 51)
4. To graph all the search results, choose **Graph search**

   or

   Select a namespace to view the metrics from that namespace. You can then 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 refine your search, hover over a metric name and choose **Add to search** or **Search for this only**.
   c. To view one of the resources on its console, choose the resource ID and then choose **Jump to resource**.
   d. To view help for a metric, select the metric name and choose **What is this?**.

   The selected metrics appear on the graph.



5. (Optional) Select one of the buttons in the search bar to edit that part of the search term.

Getting 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

Getting Statistics for a Specific Resource

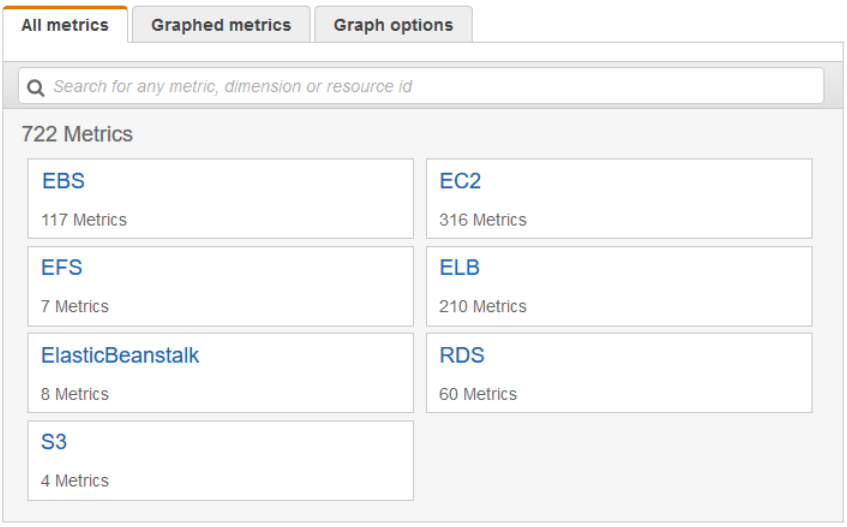
The following example shows you how to determine the maximum CPU utilization of a specific EC2 instance.

Requirements

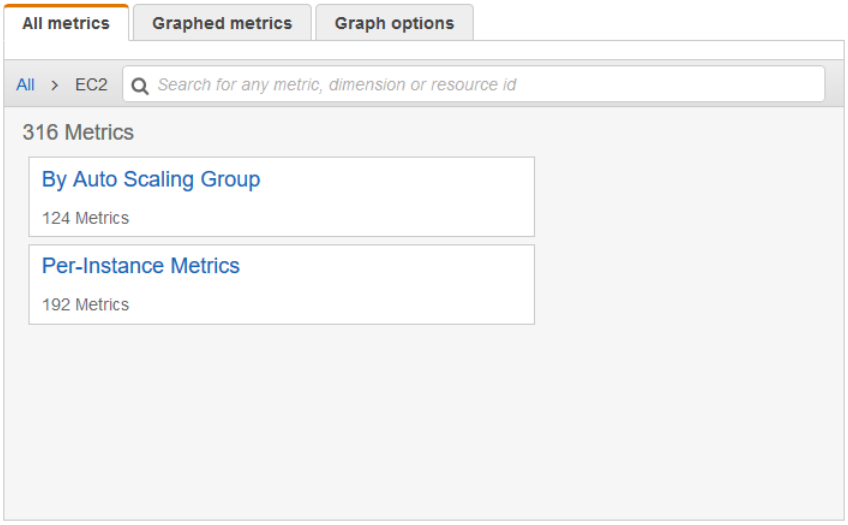
- You must have the ID of the instance. You can get the instance ID using the Amazon EC2 console or the describe-instances command.
- By default, basic monitoring is enabled, but you can enable detailed monitoring. For more information, see Enable or Disable Detailed Monitoring for Your Instances in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

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

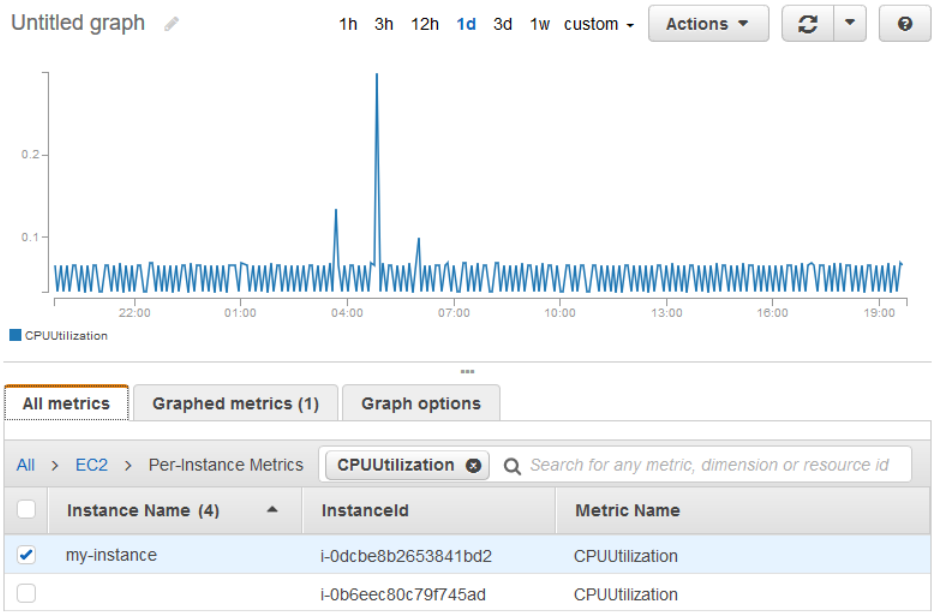
1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the **EC2** metric namespace.



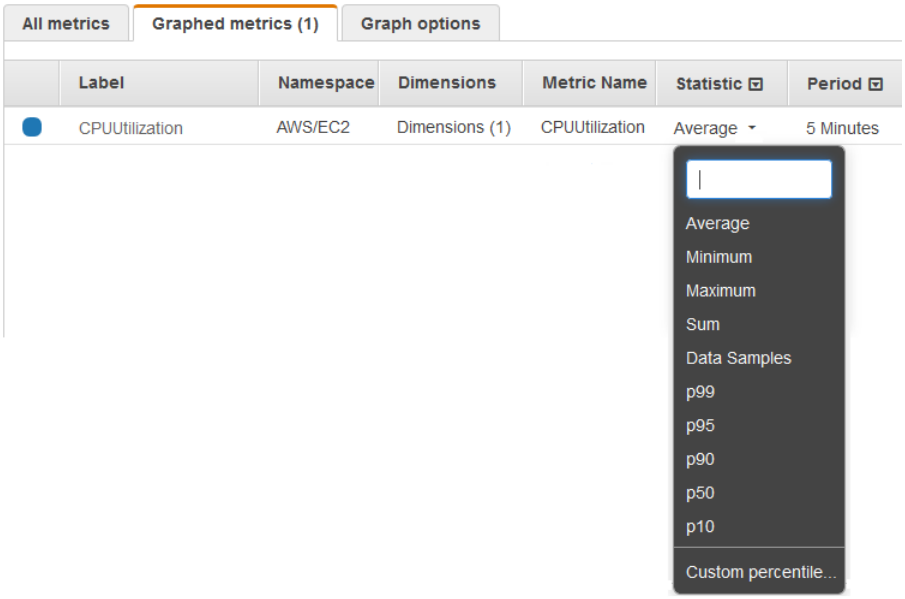
4. Select the **Per-Instance Metrics** dimension.



5. In the search field, enter **`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 6-minute values for the requested 24-hour time interval. Each value represents the maximum CPU utilization percentage for the specified instance for a particular 6-minute time period. The data points aren't 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"
```

```
}
```

Aggregating Statistics Across Resources

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

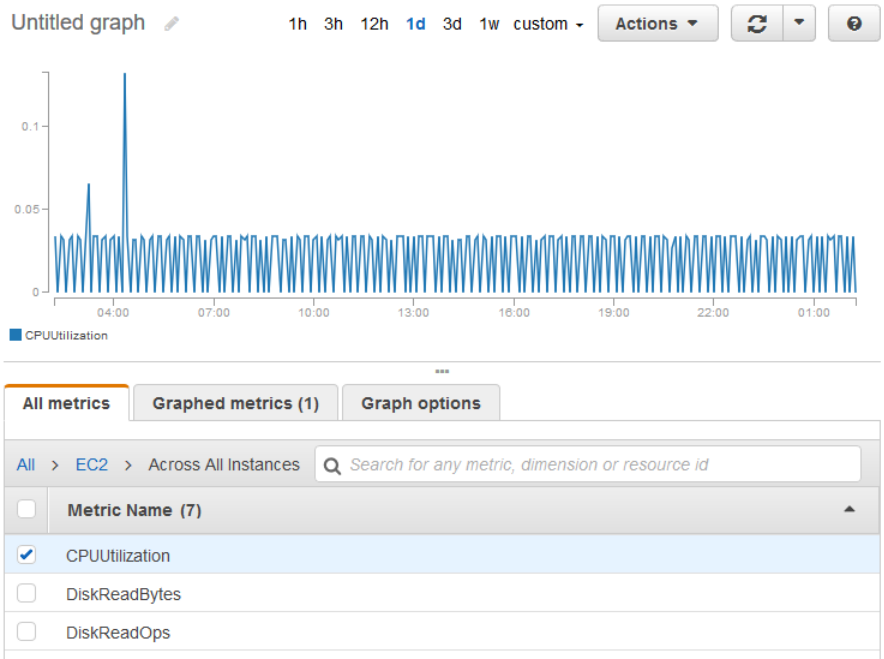
For example, you can aggregate statistics for your EC2 instances that have detailed monitoring enabled. Instances that use basic monitoring aren't included. Therefore, you must enable detailed monitoring (at an additional charge), which provides data in 1-minute periods. For more information, see Enable 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 AWS Services That Publish CloudWatch Metrics (p. 162).

> **Important**
> This technique for retrieving all dimensions across an AWS namespace doesn't work for custom namespaces that you publish to 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"
}
```

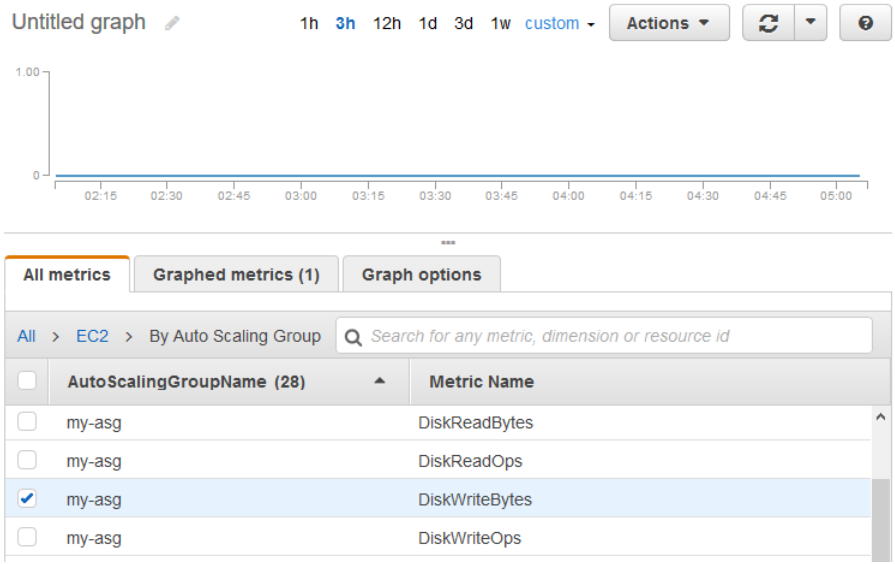
Aggregating Statistics by Auto Scaling Group

You can aggregate statistics for the EC2 instances in an Auto Scaling group. Amazon CloudWatch can't 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 1-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"
}
```

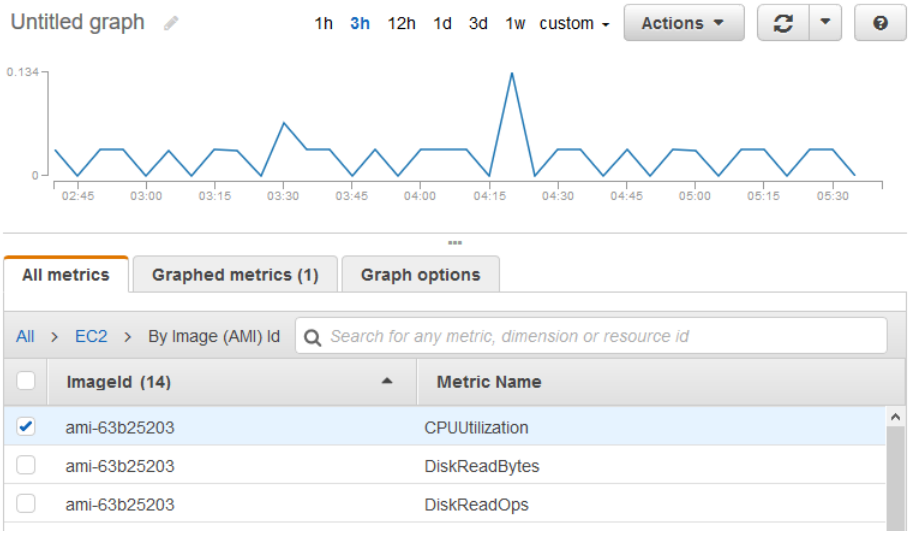
Aggregating Statistics by Amazon Machine Image (AMI)

You can aggregate statistics for the EC2 instances that have detailed monitoring enabled. Instances that use basic monitoring aren't 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-hour 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"
}
```

Graphing 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

Graphing 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).

Creating 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, enter 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** and then **Add horizontal annotation**:

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

   b. For **Value**, enter 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**.

Updating 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 Modifying the Time Range or Time Zone Format for a Graph (p. 41).
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** and then **Add horizontal annotation**:

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

   b. For **Value**, enter 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.

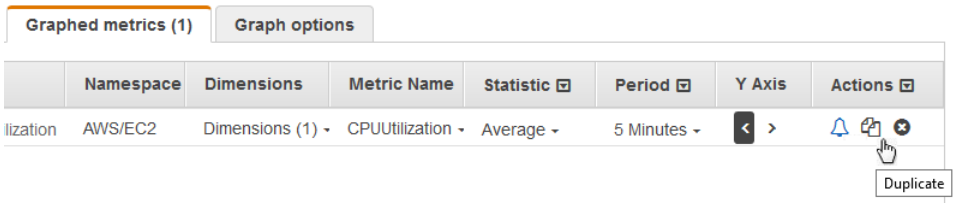
To delete an annotation, choose **x** in the **Actions** column.

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**.

Duplicating 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.

Modifying the Time Range or Time Zone Format for a Graph

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

Setting a Relative Time Range

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.

Setting an Absolute Time Range

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**.

Zooming 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.

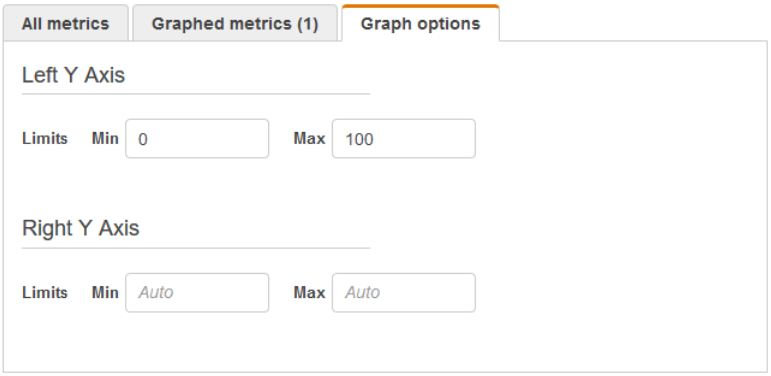
Modifying the Y-Axis for a Graph

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).

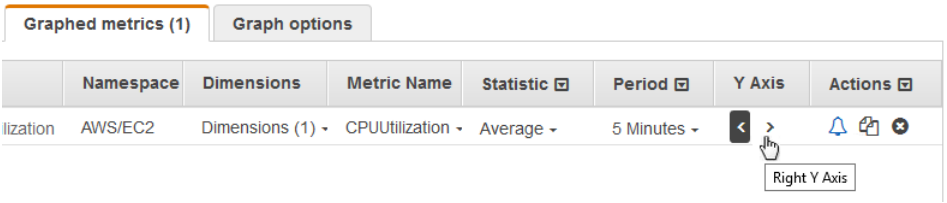
You can switch between two different y-axes for your graph. This is useful if the graph contains metrics that have different units or that differ greatly in their range of values.

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** can't 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**.

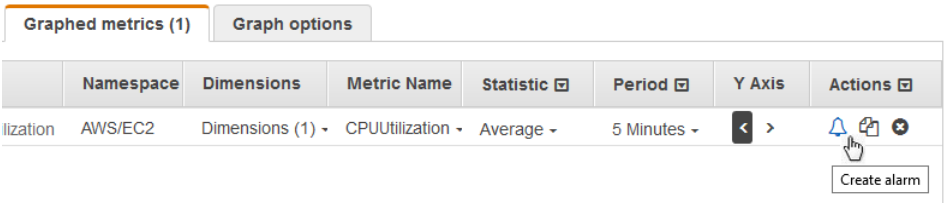


Creating an Alarm from a Metric on a Graph

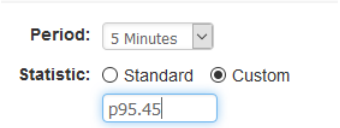
You can graph a metric and then create an alarm from the metric on the graph, which has the benefit of populating many of the alarm fields for you.

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**, enter 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**.

Publishing 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

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`*, and 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 if you have multiple dimensions, use a space between one dimension and the next.

```
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
```

To see what dimensions are defined for a metric, use the list-metrics command.

Publishing 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 2 minutes before you can retrieve statistics for the new metric using the get-metric-statistics command. However, it can take up to 15 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 1 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 1-minute period.

CloudWatch uses 1-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 1-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"
}
```

Publishing 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 3 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 can't retrieve percentile statistics for this data unless one of the following conditions is true:

- The `SampleCount` of the statistic set is 1
- The `Minimum` and the `Maximum` of the statistic set are equal

Publishing 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.

Using 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 on 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. For more information, see GetMetricData.

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, enter the math expression. The tables in the following section list the functions that 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 it 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, enter 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**. To have the legend show only the original metric labels, set ***`Expression-Label`*** to be empty.

   When one expression produces an array of time series on the graph, you can't 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**), and 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 can't 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 3 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 aren't 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. Enter 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 can't 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	Argumen	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())**

Function	Argumen	Return Type*	Description	Examples
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 aren't included You can use **METRICS()** within a larger expression that produces a 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.	**AVG(METRICS())** **SUM(METRICS("errors"))**
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())**

Function	Argumen	Return Type*	Description	Examples
SEARCH	Search expression	One or more TS	Returns one or more time series that match a search criteria that you specify. The **SEARCH** function enables you to add multiple related time series to a graph with one expression. The graph is dynamically updated to include new metrics that are added later and match the search criteria. For more information, see Using Search Expressions in Graphs (p. 51).	
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.

Using Search Expressions in Graphs

Search expressions are a type of math expression that you can add to CloudWatch graphs. Search expressions enable you to quickly add multiple related metrics to a graph. They also enable you to create dynamic graphs that automatically add appropriate metrics to their display, even if those metrics don't exist when you first create the graph.

For example, you can create a search expression that displays the `AWS/EC2 CPUUtilization` metric for all instances in the Region. If you later launch a new instance, the `CPUUtilization` of the new instance is automatically added to the graph.

When you use a search expression in a graph, the search finds the search expression in metric names, namespaces, dimension names, and dimension values. You can use Boolean operators for more complex and powerful searches.

Topics

CloudWatch Search Expression Syntax

A valid search expression has the following format.

```
SEARCH(' {Namespace, DimensionName1, DimensionName2, ...}, SearchTerm', 'Statistic',
 Period)
```

For example:

```
SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average', 300)
```

- The first part of the query after the word `SEARCH`, enclosed in curly braces, is the *metric schema* to be searched. The metric schema contains a metric namespace and one or more dimension names. Including a metric schema in a search query is optional. If specified, the metric schema must contain a namespace and can optionally contain one or more dimension names that are valid in that namespace.

  You don't need to use quote marks inside the metric schema unless a namespace or dimension name includes spaces or non-alphanumeric characters. In that case, you must enclose the name that contains those characters with double quotes.
- The `SearchTerm` is also optional, but a valid search must contain either the metric schema, the `SearchTerm`, or both. The `SearchTerm` usually contains one or more metric names or dimension values. The `SearchTerm` can include multiple terms to search for, by both partial match and exact match. It can also contain Boolean operators.

  The `SearchTerm` can include one or more designators, such as `MetricName=` as in this example, but using designators isn't required.

  The metric schema and `SearchTerm` must be enclosed together in a pair of single quote marks.
- The `Statistic` is the name of any valid CloudWatch statistic. It must be enclosed by single quotes. For more information, see Statistics (p. 5).
- The `Period` is the aggregation time period in seconds.

The preceding example searches the `AWS/EC2` namespace for any metrics that have `InstanceId` as a dimension name. It returns all `CPUUtilization` metrics that it finds, with the graph showing the `Average` statistic with an aggregation period of 5 minutes.

Search Expression Limits

The maximum search expression query size is 1024 characters. You can have as many as five search expressions on one graph. A graph can display as many as 100 time series.

CloudWatch Search Expressions: Tokenization

When you specify a `SearchTerm`, the search function searches for *tokens*, which are substrings that CloudWatch automatically generates from full metric names, dimension names, dimension values, and

namespaces. CloudWatch generates tokens distinguished by the camel-case capitalization in the original string. Numeric characters also serve as the start of new tokens, and non-alphanumeric characters serve as delimiters, creating tokens before and after the non-alphanumeric characters.

A continuous string of the same type of token delimiter character results in one token.

All generated tokens are in lowercase. The following table shows some examples of tokens generated.

Original String	Tokens Generated
CustomCount1	`customcount1`, `custom`, `count`, `1`
SDBFailure	`sdbfailure`, `sdb`, `failure`
Project2-trial333	`project2trial333`, `project`, `2`, `trial`, `333`

CloudWatch Search Expressions: Partial Matches

When you specify a `SearchTerm`, the search term is also tokenized. CloudWatch finds metrics based on partial matches, which are matches of a single token generated from the search term to a single token generated from a metric name, namespace, dimension name, or dimension value.

Partial match searches to match a single token are case insensitive. For example, using any of the following search terms can return the `CustomCount1` metric:

- `count`
- `Count`
- `COUNT`

However, using `couNT` as a search term doesn't find `CustomCount1` because the capitalization in the search term `couNT` is tokenized into `cou` and `NT`.

Searches can also match composite tokens, which are multiple tokens that appear consecutively in the original name. To match a composite token, the search is case sensitive. For example, if the original term is `CustomCount1`, searches for `CustomCount` or `Count1` are successful, but searches for `customcount` or `count1` aren't.

CloudWatch Search Expressions: Exact Matches

You can define a search to find only exact matches of your search term by using double quotes around the part of the search term that requires an exact match. These double-quotes are enclosed in the single-quotes used around the entire search term. For example, **`SEARCH(' {MyNamespace}, "CustomCount1" ', 'Maximum', 120)`** finds the exact string `CustomCount1` if it exists as a metric name, dimension name, or dimension value in the namespace named `MyNamespace`. However, the searches **`SEARCH(' {MyNamespace}, "customcount1" ', 'Maximum', 120)`** or **`SEARCH(' {MyNamespace}, "Custom" ', 'Maximum', 120)`** do not find this string.

You can combine partial match terms and exact match terms in a single search expression. For example, **`SEARCH(' {AWS/NetworkELB, LoadBalancer}, "ConsumedLCUs" OR flow ', 'Maximum', 120)`** returns the Elastic Load Balancing metric named `ConsumedLCUs` as well as all Elastic Load Balancing metrics or dimensions that contain the token `flow`.

Using exact match is also a good way to find names with special characters, such as non-alphanumeric characters or spaces, as in the following example.

```
SEARCH(' {"My Namespace", "Dimension@Name"}, "Custom:Name[Special_Characters" ', 'Maximum',
 120)
```

CloudWatch Search Expressions: Excluding a Metric Schema

All examples shown so far include a metric schema, in curly braces. Searches that omit a metric schema are also valid.

For example, **`SEARCH(' "CPUUtilization" ', 'Average', 300)`** returns all metric names, dimension names, dimension values, and namespaces that are an exact match for the string `CPUUtilization`. In the AWS metric namespaces, this can include metrics from several services including Amazon EC2, Amazon ECS, Amazon SageMaker, and others.

To narrow this search to only one AWS service, the best practice is to specify the namespace and any necessary dimensions in the metric schema, as in the following example. Although this narrows the search to the `AWS/EC2` namespace, it would still return results of other metrics if you have defined `CPUUtilization` as a dimension value for those metrics.

```
SEARCH(' {AWS/EC2, InstanceType} "CPUUtilization" ', 'Average', 300)
```

Alternatively you could add the namespace in the `SearchTerm` as in the following example. But in this example, the search would match any `AWS/EC2` string, even if it was a custom dimension name or value.

```
SEARCH(' "AWS/EC2" MetricName="CPUUtilization" ', 'Average', 300)
```

CloudWatch Search Expressions: Specifying Property Names in the Search

The following exact match search for `"CustomCount1"` returns all metrics with exactly that name.

```
SEARCH(' "CustomCount1" ', 'Maximum', 120)
```

But it also returns metrics with dimension names, dimension values, or namespaces of `CustomCount1`. To structure your search further, you can specify the property name of the type of object that you want to find in your searches. The following example searches all namespaces and returns metrics named `CustomCount1`.

```
SEARCH(' MetricName="CustomCount1" ', 'Maximum', 120)
```

You can also use namespaces and dimension name/value pairs as property names, as in the following examples. The first of these examples also illustrates that you can use property names with partial match searches as well.

```
SEARCH(' InstanceType=micro ', 'Average', 300)
```

```
SEARCH(' InstanceType="t2.micro" Namespace="AWS/EC2" ', 'Average', 300)
```

CloudWatch Search Expressions: Non-Alphanumeric Characters

Non-alphanumeric characters serve as delimiters, and mark where the names of metrics, dimensions, namespaces, and search terms are to be separated into tokens. When terms are tokenized, non-alphanumeric characters are stripped out and don't appear in the tokens. For example, `Network-Errors_2` generates the tokens `network`, `errors`, and `2`.

Your search term can include any non-alphanumeric characters. If these characters appear in your search term, they can specify composite tokens in a partial match. For example, all of the following searches would find metrics named either `Network-Errors-2` or `NetworkErrors2`.

```
network/errors
network+errors
network-errors
Network_Errors
```

When you're doing an exact value search, any non-alphanumeric characters used in the exact search must be the correct characters that appear in the string being searched for. For example, if you want to find `Network-Errors-2`, searching for `"Network-Errors-2"` is successful, but a search for `"Network_Errors_2"` isn't.

When you perform an exact match search, the following characters must be escaped with a backslash.

```
" \ ( )
```

For example, to find the metric name `Europe\France Traffic(Network)` by exact match, use the search term **`"Europe\\France Traffic\(Network\)"`**

CloudWatch Search Expressions: Boolean Operators

Search supports the use of the Boolean operators `AND`, `OR`, and `NOT` within the `SearchTerm`. Boolean operators are enclosed in the single quote marks that you use to enclose the entire search term. Boolean operators are case sensitive, so `and`, `or`, and `not` aren't valid as Boolean operators.

You can use `AND` explicitly in your search, such as **`SEARCH('{AWS/EC2,InstanceId} network AND packets', 'Average', 300)`**. Not using any Boolean operator between search terms implicitly searches them as if there were an `AND` operator, so **`SEARCH(' {AWS/EC2,InstanceId} network packets ', 'Average', 300)`** yields the same search results.

Use `NOT` to exclude subsets of data from the results. For example, **`SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT i-1234567890123456 ', 'Average', 300)`** returns the `CPUUtilization` for all your instances, except for the instance `i-1234567890123456`. You can also use a `NOT` clause as the only search term. For example, **`SEARCH( 'NOT Namespace=AWS ', 'Maximum', 120)`** yields all your custom metrics (metrics with namespaces that don't include `AWS`).

You can use multiple `NOT` phrases in a query. For example, **`SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT "ProjectA" NOT "ProjectB" ', 'Average', 300)`** returns the `CPUUtilization` of all instances in the Region, except for those with dimension values of `ProjectA` or `ProjectB`.

You can combine Boolean operators for more powerful and detailed searches, as in the following examples. Use parentheses to group the operators.

Both of the next two examples return all metric names containing `ReadOps` from both the EC2 and EBS namespaces.

```
SEARCH(' (EC2 OR EBS) AND MetricName=ReadOps ', 'Maximum', 120)
```

```
SEARCH(' (EC2 OR EBS) MetricName=ReadOps ', 'Maximum', 120)
```

The following example narrows the previous search to only results that include `ProjectA`, which could be the value of a dimension.

```
SEARCH(' (EC2 OR EBS) AND ReadOps AND ProjectA ', 'Maximum', 120)
```

The following example uses nested grouping. It returns Lambda metrics for `Errors` from all functions, and `Invocations` of functions with names that include the strings `ProjectA` or `ProjectB`.

```
SEARCH(' {AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average', 600)
```

CloudWatch Search Expressions: Using Math Expressions

You can use a search expression within a math expressions in a graph.

For example, **`SUM(SEARCH(' {AWS/Lambda, FunctionName}, MetricName="Errors" ', 'Sum', 300))`** returns the sum of the `Errors` metric of all your Lambda functions.

Using separate lines for your search expression and math expression might yield more useful results. For example, suppose that you use the following two expressions in a graph. The first line displays separate `Errors` lines for each of your Lambda functions. The ID of this expression is `e1`. The second line adds another line showing the sum of the errors from all of the functions.

```
SEARCH(' {AWS/Lambda, FunctionName}, MetricName="Errors" ', 'Sum', 300)
SUM(e1)
```

CloudWatch Search Expression Examples

The following examples illustrate more search expression uses and syntax. Let's start with a search for `CPUUtilization` across all instances in the Region and then look at variations.

This example displays one line for each instance in the Region, showing the `CPUUtilization` metric from the `AWS/EC2` namespace.

```
SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average', 300)
```

Changing `InstanceId` to `InstanceType` changes the graph to show one line for each instance type used in the Region. Data from all instances of each type is aggregated into one line for that instance type.

```
SEARCH(' {AWS/EC2,InstanceType} MetricName="CPUUtilization" ', 'Average', 300)
```

Removing the dimension name but keeping the namespace in the schema, as in the following example, results in a single line showing the aggregation of `CPUUtilization` metrics for all instances in the Region.

```
SEARCH(' {AWS/EC2} MetricName="CPUUtilization" ', 'Average', 300)
```

The following example aggregates the `CPUUtilization` by instance type and displays one line for each instance type that includes the string `micro`.

```
SEARCH(' {AWS/EC2,InstanceType} InstanceType=micro MetricName="CPUUtilization" ',
 'Average', 300)
```

This example narrows the previous example, changing the `InstanceType` to an exact search for t2.micro instances.

```
SEARCH(' {AWS/EC2,InstanceType} InstanceType="t2.micro" MetricName="CPUUtilization" ',
 'Average', 300)
```

The following search removes the `{metric schema}` part of the query, so the `CPUUtilization` metric from all namespaces appears in the graph. This can return quite a few results because the graph includes multiple lines for the `CPUUtilization` metric from each AWS service, aggregated along different dimensions.

```
SEARCH(' MetricName="CPUUtilization" ', 'Average', 300)
```

To narrow these results a bit, you can specify two specific metric namespaces.

```
SEARCH(' MetricName="CPUUtilization" AND ("AWS/ECS" OR "AWS/ES") ', 'Average', 300)
```

The preceding example is the only way to do a search of specific multiple namespaces with one search query, as you can specify only one metric schema in each query. However, to add more structure, you could use two queries in the graph, as in the following example. This example also adds more structure by specifying a dimension to use to aggregate the data for Amazon ECS.

```
SEARCH(' {AWS/ECS, ClusterName}, MetricName="CPUUtilization" ', 'Average', 300)
SEARCH(' {AWS/EBS}, MetricName="CPUUtilization" ', 'Average', 300)
```

The following example returns the Elastic Load Balancing metric named `ConsumedLCUs` as well as all Elastic Load Balancing metrics or dimensions that contain the token `flow`.

```
SEARCH(' {AWS/NetworkELB, LoadBalancer}, "ConsumedLCUs" OR flow ', 'Maximum', 120)
```

The following example uses nested grouping. It returns Lambda metrics for `Errors` from all functions and `Invocations` of functions with names that include the strings `ProjectA` or `ProjectB`.

```
SEARCH(' {AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average', 600)
```

The following example displays all of your custom metrics, excluding metrics generated by AWS services.

```
SEARCH(' NOT Namespace=AWS ', 'Average', 120)
```

The following example displays metrics with metric names, namespaces, dimension names, and dimension values that contain the string `Errors` as part of their name.

```
SEARCH(' Errors ', 'Average', 300)
```

The following example narrows that search to exact matches. For example, this search finds the metric name `Errors` but not metrics named `ConnectionErrors` or `errors`.

```
SEARCH(' "Errors" ', 'Average', 300)
```

The following example shows how to specify names that contain spaces or special characters in the metric schema part of the search term.

```
SEARCH(' {"Custom-Namespace", "Dimension Name With Spaces"}, ErrorCount ', 'Maximum', 120)
```

Creating a CloudWatch Graph with a Search Expression

On the CloudWatch console, you can access search capability when you add a graph to a dashboard, or by using the **Metrics** view.

To add a graph with a search expression to an existing 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. On the **Graphed metrics** tab, choose **Add a math expression**.
6. For **Details**, enter the search expression that you want. For example, **`SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average', 300)`**
7. (Optional) To add another search expression or math expression to the graph, choose **Add a math expression**
8. (Optional) To add a single metric to the graph, choose the **All metrics** tab and drill down to the metric you want.
9. (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**.
10. (Optional) Horizontal annotations 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** and then **Add horizontal annotation**:

    a. For **Label**, enter a label for the annotation.
    b. For **Value**, enter 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.
11. (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** and then **Add vertical annotation**:

    a. For **Label**, enter 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.

12. Choose **Create widget**.
13. Choose **Save dashboard**.

To use the Metrics view to graph searched metrics

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. In the search field, enter the tokens to search for: for example, **`cpuutilization t2.small`**.

   Results that match your search appear.
4. To graph all of the metrics that match your search, choose **Graph search**.

   or

   To refine your search, choose one of the namespaces that appeared in your search results.
5. If you selected a namespace to narrow your results, 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 refine your search, hover over a metric name and choose **Add to search** or **Search for this only**.
   c. To view help for a metric, select the metric name and choose **What is this?**.

   The selected metrics appear on the graph.
6. (Optional) Select one of the buttons in the search bar to edit that part of the search term.
7. (Optional) To add the graph to a dashboard, choose **Actions** and then **Add to dashboard**.

Using Amazon CloudWatch Alarms

You can create a CloudWatch alarm that watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. The alarm performs one or more actions based on the value of the metric or expression 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 or expression is within the defined threshold.
- *`ALARM`*—The metric or expression 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.

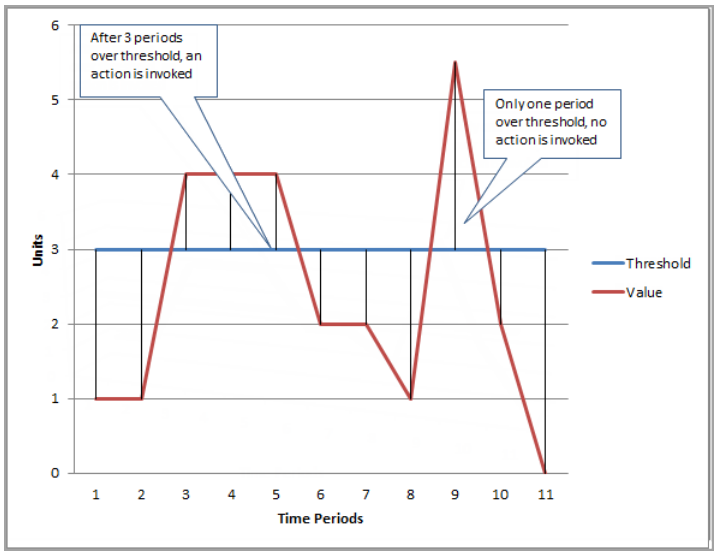
Evaluating an Alarm

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 or expression to create each individual data point for an alarm. 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 periods, or 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 that something 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 the 3 necessary data points are missing. Even though the most recent 5 data points are evaluated, only 3 (the setting for **Evaluation Periods**) are necessary to evaluate the alarm state. The number of data points in Column 2 is the number of data points that must be "filled in", using the setting for how missing data is being treated.

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
- - - - -	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 Publishing Custom Metrics (p. 44).

Alarms on Math Expressions

You can set an alarm on the result of a math expression that is based on one or more CloudWatch metrics. A math expression used for an alarm can include as many as 10 metrics. Each metric must be using the same period.

For an alarm based on a math expression, you can specify how you want CloudWatch to treat missing data points for the underlying metrics when evaluating the alarm.

Alarms based on math expressions cannot perform Amazon EC2 actions.

For more information about metric math expressions and syntax, see Using Metric Math (p. 47).

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 timestamp. In rare cases, your history might show more than one notification for a state change. The timestamp 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, 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, 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 a CloudWatch Alarm Based on a CloudWatch Metric

You choose a CloudWatch metric to for the alarm to watch, and the threshold for that metric. The alarm goes to ALARM state when the metric breaches the threshold for a specified number of evaluation periods.

To create an alarm based on a single metric

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric** and do one of the following:

   a. Choose the service namespace that contains the metric you want. Continue choosing options as they appear to narrow the choices. When a list of metrics is displayed, select the checkbox next to the metric you want.

   or

   b. In the search box, type the name of a metric, dimension, or resource ID and press ENTER. Then choose one of the results and continue until a list of metrics is displayed. Select the checkbox next to the metric you want.
4. Choose the **Graphed metrics** tab.

   a. Under **Statistic** , choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).

   b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point.

   You can also choose whether the Y axis legend appears on the left or right while you are creating the alarm. This preference is used only while you are creating the alarm.

   c. Choose **Select metric**.
5. Type a name and description for the alarm. The name must contain only ASCII characters.
6. For **Whenever**, specify the alarm condition.

a. For **is:**, specify whether the metric must be greater than, less than, or equal to the threshold, and specify the threshold value.

b. For **for:**, specify how many evaluation periods (datapoints) must be in the ALARM state to trigger the alarm. Initially, you can change only the second value, and the first value changes to match your entry. This creates an alarm that goes to ALARM state if that many consecutive periods are breaching.

   To create an M out of N alarm, choose the pencil icon. You can then change the M number to be different than the N number. For more information, see Evaluating an Alarm (p. 60).

7. Under **Additional settings**, for **Treat missing data as**, choose how to have the alarm behave when some data points are missing. For more information, see Configuring How CloudWatch Alarms Treat Missing Data (p. 61).
8. If the alarm uses a percentile as the monitored statistic, a **Percentiles with low samples** box is displayed. Use it to choose whether to evaluate or ignore cases with low sample rates. If you choose **ignore (maintain the alarm state)**, the current alarm state is always maintained when the sample size is too low. For more information, see Percentile-Based CloudWatch Alarms and Low Data Samples (p. 64).
9. Under **Actions**, select the type of action to have the alarm to perform when the alarm is triggered. Use the **+Notification**, **+AutoScaling Action**, and **+EC2 Action** buttons to have the alarm perform multiple actions. Specify at least one action.
10. 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. 24).

Create a CloudWatch Alarm Based on a Metric Math Expression

To create an alarm based on a metric math expression, choose one or more CloudWatch metrics to use in the expression. Then, specify the expression, threshold, and evaluation periods.

To create an alarm based on a math expression

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric** and do one of the following:

   a. Choose the service namespace that contains a specific metric. Continue choosing options as they appear to narrow the choices. When a list of metrics is displayed, select the check box next to the right metric.

      or

   b. In the search box, type the name of a metric, dimension, or resource ID and press Enter. Choose one of the results and continue until a list of metrics is displayed. Select the check box next to the right metric.

   (Optional) To add another metric to use in the math expression, under **All metrics**, choose **All**, find the specific metric, and then select the check box next to it. You can add up to 10 metrics.
4. Choose **Graphed metrics**. For each metric added, do the following:

a. Under **Statistic**, choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, p95.45).

b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point.

You can also choose whether the Y axis legend appears on the left or right while you are creating the alarm. This preference is used only while you are creating the alarm.

5. Choose **Add a math expression**. A new row appears for the expression.
6. On the new row, under the **Details** column, type the math expression and press Enter. For information about the functions and syntax that you can use, see Metric Math Syntax and Functions (p. 47).

   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 underscores, and must start with a lowercase letter. Changing the value of **Id** to a more meaningful name can also make the alarm graph easier to understand.
7. (Optional), add more math expressions, using both metrics and the results of other math expressions in the formulas of the new math expressions.
8. When you have the expression to use for the alarm, clear the check boxes to the left of every other expression and every metric on the page. Only the check box next to the expression to use for the alarm should be selected. The expression you choose for the alarm must produce a single time series, and show only one line on the graph. Then choose **Select metric**.
9. Type a name and description for the alarm. The name must contain only ASCII characters.
10. For **Whenever**, specify the alarm condition.

    a. For **is:**, specify whether the expression result must be greater than, less than, or equal to the threshold, and specify the threshold value.

    b. For **for:**, specify how many evaluation periods (data points) must be in the ALARM state to trigger the alarm. Initially, you can change only the second value, and the first value changes to match your entry. This creates an alarm that goes to the ALARM state if that many consecutive periods are breaching.

    To create an M out of N alarm, choose the pencil icon. You can then change the M number to be different than the N number. For more information, see Evaluating an Alarm (p. 60).
11. Under **Additional settings**, for **Treat missing data as**, choose how to have the alarm behave when some data points are missing. For more information, see Configuring How CloudWatch Alarms Treat Missing Data (p. 61).
12. Under **Actions**, select the type of action to have the alarm to perform when the alarm is triggered. Choose **+Notification** or **+AutoScaling Action** to have the alarm perform multiple actions. Specify at least one action.
13. 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. 24).

Edit a CloudWatch Alarm

You can edit an existing alarm, and update the Amazon SNS email notification list it uses.

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 choose **Actions**, **Modify**.
4. On the **Modify Alarm** page, update the alarm as necessary and choose **Save Changes**. To modify the expression, metrics, period, or statistic, first choose **Edit** near the top of the screen.

   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.

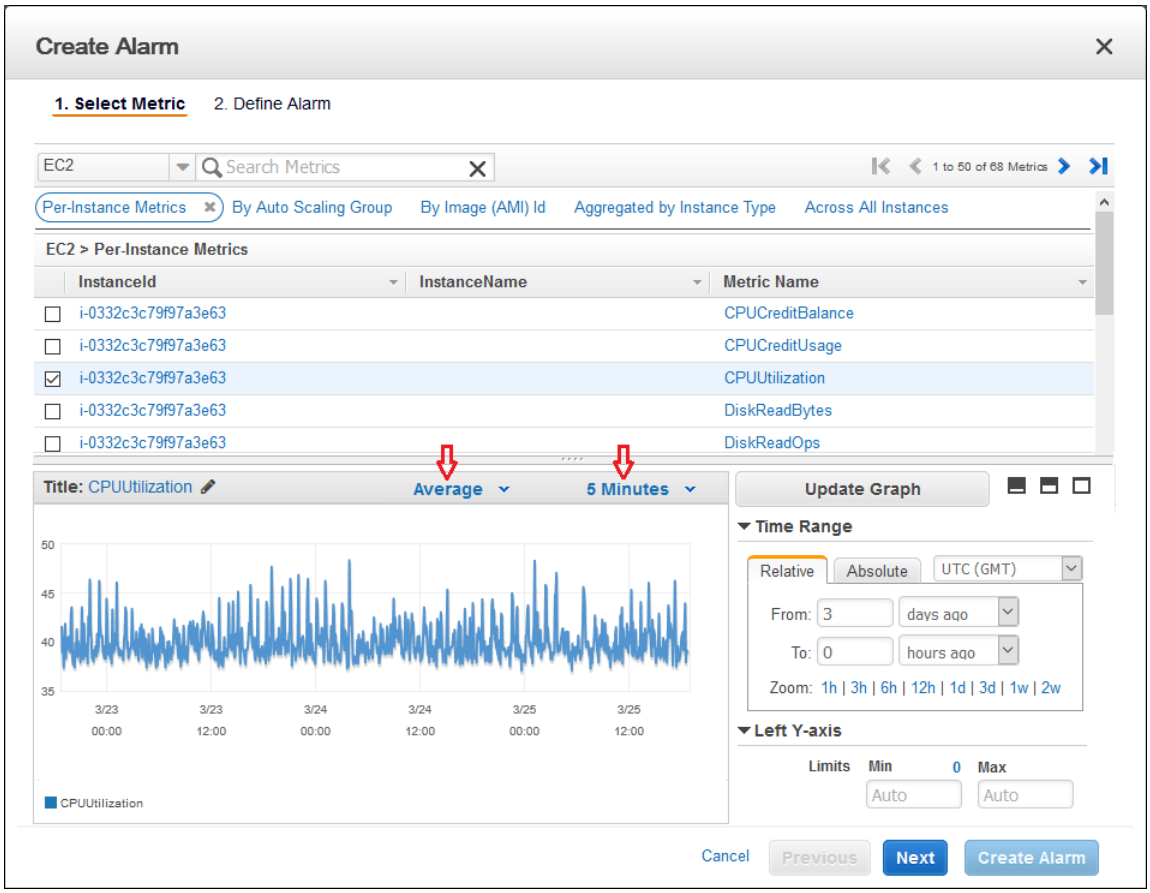
Set Up a CPU Usage Alarm Using the AWS Management Console

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:

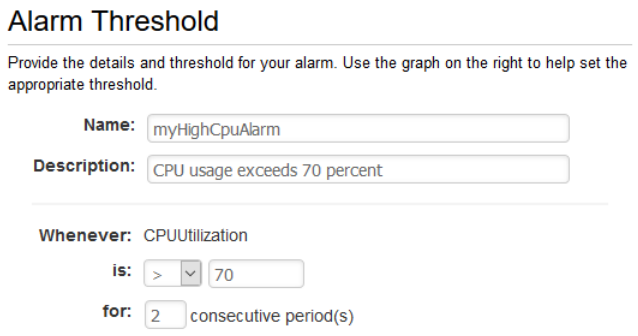
   a. Select a row with the instance and the **CPUUtilization** metric.
   b. For the statistic, choose **Average**, choose one of the predefined percentiles, or specify a custom percentile (for example, p95.45).
   c. Choose a period (for example, **`5 minutes`**).
   d. Choose **Next**.



5. Define the alarm as follows:

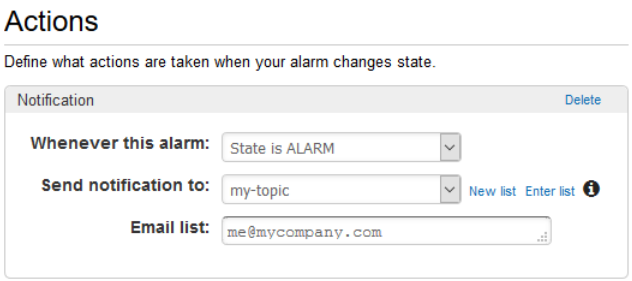
   a. Under **Alarm Threshold**, type a unique name for the alarm (for example, myHighCpuAlarm) and a description of the alarm (for example, CPU usage exceeds 70 percent). Alarm names must contain only ASCII characters.

   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. 65).
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. 65).
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 100 MB 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 Set Up Amazon SNS Notifications (p. 65).
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 helps 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. You can also 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 aren't 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 are 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

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 supported only on:

- The A1, C3, C4, C5, C5n, M3, M4, M5, M5a, R3, R4, R5, R5a, T2, T3, X1, and X1e 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

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.

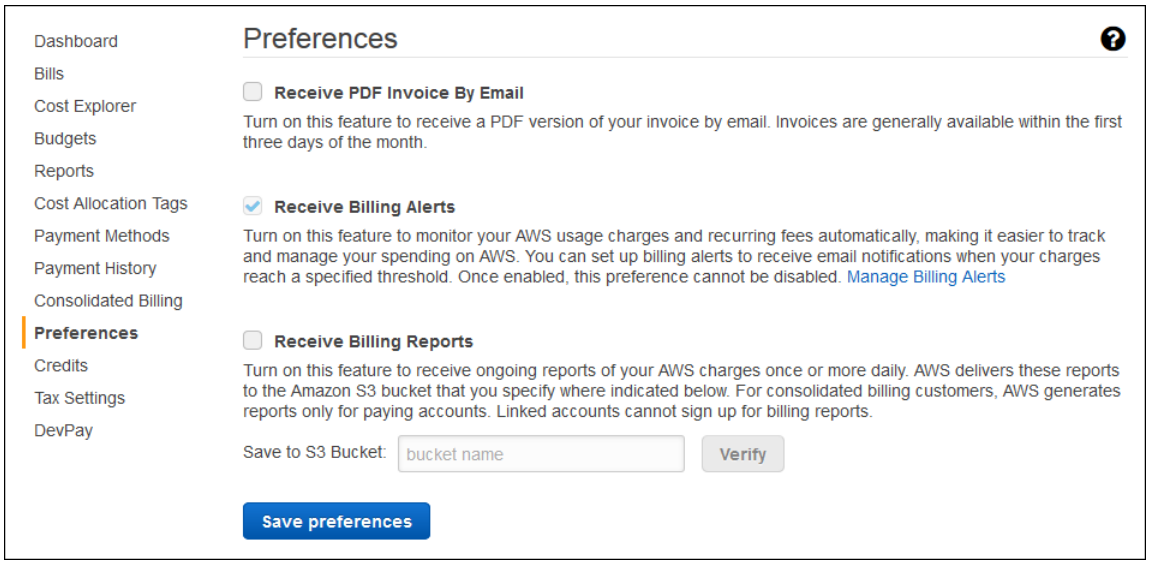
After you enable billing alerts for the first time, it takes about 15 minutes before you can view billing data and set billing alarms.

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. 173) 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**.

Collecting 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 across operatins systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances. The additional metrics that can be collected are listed in Metrics Collected by the CloudWatch Agent (p. 143).
- Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
- 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.
- Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.

You can store and view the metrics that 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 Amazon 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 the CloudWatch agent can collect, see Metrics Collected by the CloudWatch Agent (p. 143).

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 versions 18.04, 16.04, and 14.04
- CentOS versions 7.6, 7.0, and 6.5
- Red Hat Enterprise Linux (RHEL) version 7.5, 7.4, 7.2, 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

You can download and install the CloudWatch agent manually using the command line, or you can integrate it with SSM. The general flow of installing the CloudWatch agent using either method is as follows:

1. Create IAM roles or users that enable the agent to collect metrics from the server and optionally to integrate with AWS Systems Manager.
2. Download the agent package.
3. Modify the CloudWatch agent configuration file and specify the metrics that you want to collect.
4. Install and start the agent on your servers. As you install the agent on an EC2 instance, you attach the IAM role that you created in step 1. As you install the agent on an on-premises server, you specify a named profile that contains the credentials of the IAM user that you created in step 1.

Contents

Installing the CloudWatch Agent

You can download and install the CloudWatch agent using either the command line with an Amazon S3 download link, using SSM, or using an AWS CloudFormation template.

Contents

Installing the CloudWatch Agent Using the Command Line

Use the following topics to download, configure, and install the CloudWatch agent package.

Topics

Download and Configure the CloudWatch Agent Using the Command Line

Use the following steps to download the CloudWatch agent package, create IAM roles or users, and optionally modify the common configuration file.

Download the CloudWatch Agent Package Using an S3 Download Link

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

Architecture	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/amazon-cloudwatch-agent.msi	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig
arm64	Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
arm64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
arm64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/	https://s3.amazonaws.com/amazoncloudwatch-agent/

Architecture	Platform	Download Link	Signature File Link
		ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig

To use the command line to download and install the CloudWatch agent package

1. Download the CloudWatch agent.

   On a Linux server, enter the following. For *download-link*, use the appropriate download link from the previous table.

   ```
   wget download-link
   ```

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

   ```
   https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi
   ```

2. After you have downloaded the package, you can optionally use a GPG signature file to verify the package signature. For more information, see Verifying the Signature of the CloudWatch Agent Package (p. 108).
3. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and enter the following:

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

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

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

   If you downloaded an MSI package on a server running Windows Server, change to the directory containing the package and enter the following:

   ```
   msiexec /i amazon-cloudwatch-agent.msi
   ```

   This command also works from within PowerShell. For more information about MSI command options, see Command-Line Options in the Microsoft Windows documentation.

Create and Modify the Agent Configuration File

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. 111).

Create IAM Roles and Users for Use With CloudWatch Agent

Access to AWS resources requires permissions. You create an IAM role, an IAM user, or both to grant permissions that the CloudWatch agent needs to write metrics to CloudWatch. If you're going to use the agent on Amazon EC2 instances, you must create an IAM role. If you're going to use the agent on on-premises servers, you must create an IAM user.

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 file name that doesn't start with `AmazonCloudWatch-`, these policies can't be used to write the file to Parameter Store or download it from Parameter Store.

If you're going to run the CloudWatch agent on Amazon EC2 instances, use the following steps to create the necessary IAM role. This role provides permissions for reading information from the instance and writing it to CloudWatch.

To create the IAM role necessary to run the CloudWatch agent on EC2 instances

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** and then **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**. If necessary, use the search box to find the policy.
5. Choose **Next: Review**.
6. Confirm that **CloudWatchAgentServerPolicy** appears next to **Policies**. In **Role name**, enter a name for the role, such as *`CloudWatchAgentServerRole`*. Optionally give it a description. Then choose **Create role**.

   The role is now created.

If you're going to run the CloudWatch agent on on-premises servers, use the following steps to create the necessary IAM user.

To create the IAM user necessary for the CloudWatch agent to run on on-premises servers

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** and then **Add user**.
3. Enter 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**. If necessary, use the search box to find the policy.
7. Choose **Next: Review**.
8. Confirm that the correct policies are listed, and choose **Create user**.
9. 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. Choose **Close**.

Installing and Running the CloudWatch Agent on Your Servers

After you have created the agent configuration file that you want and created an IAM role or IAM user, use the following steps to install and run the agent on your servers, using that configuration. First, attach

an IAM role or IAM user to the server that will run the agent. Then, on that server, download the agent package and start it using the agent configuration you created.

Download the CloudWatch Agent Package Using an S3 Download Link

On each server where you will run the agent, download the agent package. Choose the download link from this table, depending on your architecture and platform.

Architecture	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/amazon-cloudwatch-agent.msi	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig
arm64	Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
arm64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig

Architecture	Platform	Download Link	Signature File Link
arm64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig

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

1. Download the CloudWatch agent. For a Linux server, enter the following. For *download-link*, use the appropriate download link from the previous table.

   ```
   wget download-link
   ```

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

   ```
   https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi
   ```

2. After you have downloaded the package, you can optionally use a GPG signature file to verify the package signature. For more information, see Verifying the Signature of the CloudWatch Agent Package (p. 108).
3. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and enter the following:

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

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

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

   If you downloaded an MSI package on a server running Windows Server, change to the directory containing the package and enter the following:

   ```
   msiexec /i amazon-cloudwatch-agent.msi
   ```

   This command also works from within PowerShell. For more information about MSI command options, see Command-Line Options in the Microsoft Windows documentation.

(Installing on an EC2 Instance) Attaching an IAM Role

To enable the CloudWatch agent to send data from the instance, you must attach an IAM role to the instance. Use the role that you created earlier. For more information about creating this role, see Create IAM Roles and Users for Use with the CloudWatch Agent (p. 93).

We recommended `CloudWatchAgentServerRole` as the name when you created the role. In the IAM console, you can scroll through the list to find it, or you can use the search box.

For more information on attaching an IAM role to an instance, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide for Windows Instances*.

(Installing on an On-Premises Server) Specify IAM Credentials and AWS Region

To enable the CloudWatch agent to send data from an on-premises server, you must specify the access key and secret key of the IAM user that you created earlier. For more information about creating this user, see Create IAM Roles and Users for Use with the CloudWatch Agent (p. 93).

You also must specify the AWS Region to send the metrics to, using the `region` field in the `[AmazonCloudWatchAgent]` section of the AWS config file, as in the following example.

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

The 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

- On Linux servers, enter the following command and follow the prompts:

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

  On Windows Server, open PowerShell as an administrator, enter the following command, and follow the prompts.

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

(Optional) Modify the Common Configuration for Proxy or Region Information

The CloudWatch agent includes a configuration file called `common-config.toml`. You can optionally use this file to specify proxy 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` – For on-premises servers, this line specifies the IAM user credential profile to use to send data to CloudWatch. If you keep this line commented out, `AmazonCloudWatchAgent` is used. For more information about creating this profile, see (Installing on an On-Premises Server) Specify IAM Credentials and AWS Region (p. 91).

  On an EC2 instance, you can use this line to have the CloudWatch agent send data from this instance to CloudWatch in a different AWS Region. To do so, specify a named profile that includes a `region` field specifying the name of the Region to send to.
- `shared_credential_file` – To have the agent look for credentials in a file located in a path other than the default path, specify that complete path and file name here.
- 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.

Start the CloudWatch Agent Using the Command Line

Follow these steps to use the command line to start the CloudWatch agent on a server.

To use the command line to start the CloudWatch agent on a server

1. Copy the agent configuration file that you want to use to the server where you're going to run the agent. Note the pathname where you copy it to.
2. 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.

   On an EC2 instance running Linux, enter the following:

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

   On an on-premises server running Linux, enter the following:

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

   On an EC2 instance running Windows Server, enter the following from the PowerShell console:

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

   On an on-premises server running Windows Server, enter the following from the PowerShell console:

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

Installing the CloudWatch Agent Using SSM

Use the following topics to install and run the CloudWatch agent using SSM.

Topics

Create IAM Roles and Users for Use with the CloudWatch Agent

Access to AWS resources requires permissions. You can create IAM roles and users that include the permissions that 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.

One role or user enables 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. You should use it only when you need it, and it shouldn't be attached to multiple instances in your deployment. If you're going to store your CloudWatch agent configuration in Parameter Store, you should set up one instance where you perform this configuration. You should 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 file name that doesn't start with `AmazonCloudWatch-`, these policies can't be used to write the file to Parameter Store or download it from Parameter Store.

Creat IAM Roles to Use with the 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're 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 the 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** and then **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**. If necessary, use the search box to find the policy.

5. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. If necessary, use the search box to find the policy. This policy isn't 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**, enter a name for the role, such as *`CloudWatchAgentServerRole`*. Optionally give it a description. Then 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're going to store the agent configuration file in Parameter Store so that other servers can retrieve it.

The permissions for writing to Parameter Store provide broad powers. This role shouldn't be attached to all your servers, and only administrators should use it. After you're finished creating the agent configuration file and copying it to Parameter Store, you should detach this role from the instance and use `CloudWatchAgentServerRole` 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** and then **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**. If necessary, use the search box to find the policy.
5. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. If necessary, use the search box to find the policy. This policy isn't 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**, enter a name for the role, such as *`CloudWatchAgentAdminRole`*. Optionally give it a description. Then choose **Create role**.

   The role is now created.

Create IAM Users to Use with the 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 the 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** and then **Add user**.

3. Enter 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**. If necessary, use the search box to find the policy.
7. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. If necessary, use the search box to find the policy. This policy isn't 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. Choose **Close**.

The following procedure creates the IAM user that can also write to Parameter Store. If you're 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. This user shouldn't be attached to all your servers, and only administrators should use it. 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** and then **Add user**.
3. Enter 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**. If necessary, use the search box to find the policy.
7. To use SSM to install or configure the CloudWatch agent, select the check box next to **AmazonEC2RoleforSSM**. If necessary, use the search box to find the policy. This policy isn't 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. Choose **Close**.

Download and Configure the CloudWatch Agent

This section explains how to use Systems Manager to download the agent and then how to create your agent configuration file. Before you can use Systems Manager to download the agent, you must make sure that the instance is configured correctly for Systems Manager.

Installing or Updating 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 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're 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're 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're 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*.

Use the following steps to download the CloudWatch agent package using Systems Manager.

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 to install the CloudWatch agent on. If you don't see a specific instance, it might not be configured for Run Command. For more information, see Systems Manager Prerequisites in the *AWS Systems Manager User Guide*.
6. In the **Action** list, choose **Install**.
7. In the **Name** field, enter `AmazonCloudWatchAgent`.
8. Keep **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.

Create and Modify the Agent Configuration File

After you have downloaded the CloudWatch agent, you must create the configuration file before you start the agent on any servers.

If you're going to save your agent configuration file in the Systems Manager Parameter Store, you must use an EC2 instance to save to the Parameter Store. Additionally, you must first attach to that instance the IAM role that you created that has permissions to write to Parameter Store. This role might be called `CloudWatchAgentAdminRole`. For more information about attaching roles, see Attaching an IAM Role to an Instance in the *Amazon EC2 User Guide for Windows Instances*.

For more information about creating the CloudWatch agent configuration file, see Create the CloudWatch Agent Configuration File (p. 111).

Installing the CloudWatch Agent on EC2 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

Attach an IAM Role to the Instance

You must attach an IAM role to the EC2 instance to be able to run the CloudWatch agent on the 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 might be called `CloudWatchAgentServerPolicy`.

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. For information about using an Amazon S3 download link, see Download the CloudWatch Agent Package Using an S3 Download Link (p. 86).

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.

Installing or Updating 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 *AWS Systems Manager User Guide*.
6. In the **Action** list, choose **Install**.
7. In the **Name** box, enter `AmazonCloudWatchAgent`.
8. Keep **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.

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

The CloudWatch agent includes a configuration file called `common-config.toml`. You can use this file optionally specify proxy 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` – For on-premises servers, this line specifies the IAM user credential profile to use to send data to CloudWatch. If you keep this line commented out, `AmazonCloudWatchAgent` is used.

  On an EC2 instance, you can use this line to have the CloudWatch agent send data from this instance to CloudWatch in a different AWS Region. To do so, specify a named profile that includes a `region` field specifying the name of the Region to send to.
- `shared_credential_file` – To have the agent look for credentials in a file located in a path other than the default path, specify that complete path and file name here. If you use this option, be sure that the `shared_credential_profile` option is commented out.
- 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.

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, enter 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. 111).
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: If you saved the configuration file in the Systems Manager Parameter Store, enter the following:

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

  Linux: If you saved the configuration file on the local computer, enter the following:

  ```
  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, enter the following from the PowerShell console:

  ```
  ./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, enter the following from the PowerShell console:

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

Installing the CloudWatch Agent on On-Premises Servers

If you have downloaded the CloudWatch agent on one computer and created the agent configuration file you want, you can use that configuration file to install the agent on other on-premises servers.

Download the CloudWatch Agent on an On-Premises Server

You can download the CloudWatch agent package using either Systems Manager Run Command or an Amazon S3 download link. For information about using an Amazon S3 download link, see Download the CloudWatch Agent Package Using an S3 Download Link (p. 86).

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 to install the CloudWatch agent on. If you don't see a specific server, it might not be configured for Run Command. For more information, see Systems Manager Prerequisites in the *AWS Systems Manager User Guide*.
6. In the **Action** list, choose **Install**.
7. In the **Name** box, enter *`AmazonCloudWatchAgent`*.
8. Keep **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.

(Installing on an On-Premises Server) Specify IAM Credentials and AWS Region

To enable the CloudWatch agent to send data from an on-premises server, you must specify the access key and secret key of the IAM user that you created earlier. For more information about creating this user, see Create IAM Roles and Users for Use with the CloudWatch Agent (p. 93).

You also must specify the AWS Region to send the metrics to, using the `region` field.

Following is an example of this file.

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
region = us-west-1
```

For *`my_access_key`* and *`my_secret_key`*, use the keys from the IAM user that doesn't have the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for CloudWatch agent, see Create IAM Users to Use with the CloudWatch Agent on On-Premises Servers (p. 94).

If you name this profile `AmazonCloudWatchAgent`, you don't need to do anything more. Optionally, you can give it a different name and specify that name as the value for `shared_credential_profile` in the `common-config.toml` file, which is explained in the following section.

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

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

- On Linux servers, enter the following command and follow the prompts:

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

On Windows Server, open PowerShell as an administrator, enter the following command, and follow the prompts.

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

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

The CloudWatch agent includes a configuration file called `common-config.toml`. You can optionally use this file to specify proxy 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` – For on-premises servers, this line specifies the IAM user credential profile to use to send data to CloudWatch. If you keep this line commented out, `AmazonCloudWatchAgent` is used. For more information about creating this profile, see (Installing on an On-Premises Server) Specify IAM Credentials and AWS Region (p. 101).

  On an EC2 instance, you can use this line to have the CloudWatch agent send data from this instance to CloudWatch in a different AWS Region. To do so, specify a named profile that includes a `region` field specifying the name of the Region to send to.
- `shared_credential_file` – To have the agent look for credentials in a file located in a path other than the default path, specify that complete path and file name here. If you use this option, be sure that the `shared_credential_profile` option is commented out.

- 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.

Starting 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, enter 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: If you saved the configuration file in the Systems Manager Parameter Store, enter the following:

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

  Linux: If you saved the configuration file on the local computer, enter the following:

```
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, enter the following from the PowerShell console:

```
./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, enter the following from the PowerShell console:

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

Installing the CloudWatch Agent Using AWS CloudFormation

Amazon has uploaded several AWS CloudFormation templates to GitHub to help you install and update the CloudWatch agent. For more information about using AWS CloudFormation, see What is AWS CloudFormation?.

The template location is Deploy the Amazon CloudWatch agent to EC2 instances using AWS CloudFormation. This location includes both `inline` and `ssm` directories. Each of these directories contains templates for both Linux and Windows instances.

- The templates in the `inline` directory have the CloudWatch agent configuration embedded into the AWS CloudFormation template. By default, the Linux templates collect the metrics `mem_used_percent` and `swap_used_percent`, and the Windows templates collect `Memory % Committed Bytes In Use` and `Paging File % Usage`.

  To modify these templates to collect different metrics, modify the following section of the template. The following example is from the template for Linux servers. Follow the format and syntax of the agent configuration file to make these changes. For more information, see Manually Create or Edit the CloudWatch Agent Configuration File (p. 116).

```
{
   "metrics":{
      "append_dimensions":{
         "AutoScalingGroupName":"${!aws:AutoScalingGroupName}",
         "ImageId":"${!aws:ImageId}",
         "InstanceId":"${!aws:InstanceId}",
         "InstanceType":"${!aws:InstanceType}"
      },
      "metrics_collected":{
         "mem":{
            "measurement":[
               "mem_used_percent"
            ]
         },
         "swap":{
            "measurement":[
               "swap_used_percent"
            ]
         }
      }
   }
}
```

  Note
  In the inline templates, all placeholder variables must have an exclamation mark (!) before them as an escape character. You can see this in the example template. If you add other placeholder variables, be sure to add an exclamation mark before the name.

- The templates in the `ssm` directory load an agent configuration file from Parameter Store. To use these templates, you must first create a configuration file and upload it to Parameter Store. You then provide the Parameter Store name of the file in the template. You can create the configuration file manually or by using the wizard. For more information, see Create the CloudWatch Agent Configuration File (p. 111).

You can use both types of templates for installing the CloudWatch agent and for updating the agent configuration.

Tutorial: Install and Configure the CloudWatch Agent Using an AWS CloudFormation Inline Template

This tutorial walks you through using AWS CloudFormation to install the CloudWatch agent on a new Amazon EC2 instance. This tutorial installs on a new instance running Amazon Linux 2 using the inline templates, which don't require the use of the JSON configuration file or Parameter Store. The inline template includes the agent configuration in the template. In this tutorial, you use the default agent configuration contained in the template.

After the procedure for installing the agent, the tutorial continues with how to update the agent.

To use AWS CloudFormation to install the CloudWatch agent on a new instance

1. Download the template from GitHub. In this tutorial, download the inline template for Amazon Linux 2 as follows:

   ```
   curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/master/
   aws/solutions/AmazonCloudWatchAgent/inline/amazon_linux.template
   ```

2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Choose **Create stack**.
4. For **Choose a template**, select **Upload a template to Amazon S3**, choose the downloaded template, and choose **Next**.
5. On the **Specify Details** page, fill out the following parameters and choose **Next**:

   - **Stack name**: Choose a stack name for your AWS CloudFormation stack.
   - **IAMRole**: Choose an IAM role that has permissions to write CloudWatch metrics and logs. For more information, see Creat IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances (p. 93).
   - **InstanceAMI**: Choose an AMI that is valid in the Region where you're going to launch your stack.
   - **InstanceType**: Choose a valid instance type.
   - **KeyName**: To enable SSH access to the new instance, choose an existing Amazon EC2 key pair. If you don't already have an Amazon EC2 key pair, you can create one in the AWS Management Console. For more information, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide for Linux Instances*.
   - **SSHLocation**: Specifies the IP address range that can be used to connect to the instance using SSH. The default allows access from any IP address.
6. On the **Options** page, you can choose to tag your stack resources. Choose **Next**.
7. On the **Review** page, review your information, acknowledge that the stack might create IAM resources, and then choose **Create**.

   If you refresh the console, you see that the new stack has the `CREATE_IN_PROGRESS` status.
8. When the instance is created, you can see it in the Amazon EC2 console. Optionally, you can connect to the host and check the progress.

   Use the following command to confirm that the agent is installed:

   ```
   rpm -qa amazon-cloudwatch-agent
   ```

   Use the following command to confirm that the agent is running:

```
ps aux | grep amazon-cloudwatch-agent
```

The next procedure demonstrates using AWS CloudFormation to update the CloudWatch agent using an inline template. The default inline template collects the `mem_used_percent` metric. In this tutorial, you change the agent configuration to stop collecting that metric.

To use AWS CloudFormation to update the CloudWatch agent

1. In the template that you downloaded in the previous procedure, remove the following lines and then save the template:

```
"mem": {

     "measurement": [
         "mem_used_percent"
       ]
 },
```

2. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. On the AWS CloudFormation dashboard, select the stack that you created and choose **Update Stack**.
4. For **Select Template**, select **Upload a template to Amazon S3**, choose the template that you modified, and choose **Next**.
5. On the **Options** page, choose **Next** and then **Next**.
6. On the **Review** page, review your information and choose **Update**.

   After some time, you see `UPDATE_COMPLETE`.

Tutorial: Install the CloudWatch Agent Using AWS CloudFormation and Parameter Store

This tutorial walks you through using AWS CloudFormation to install the CloudWatch agent on a new Amazon EC2 instance. This tutorial installs on a new instance running Amazon Linux 2 using an agent configuration file that you create and save in Parameter Store.

After the procedure for installing the agent, the tutorial continues with how to update the agent.

To use AWS CloudFormation to install the CloudWatch agent on a new instance using a configuration from Parameter Store

1. If you haven't done so already, download the CloudWatch agent package to one of your computers so that you can create the agent configuration file. For more information and downloading the agent using Parameter Store, see ??? (p. 95). For more information on downloading the package using the command line, see Download and Configure the CloudWatch Agent Using the Command Line (p. 85).
2. Create the agent configuration file and save it in Parameter Store. For more information, see Create the CloudWatch Agent Configuration File (p. 111).
3. Download the template from GitHub as follows:

```
curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/master/
aws/solutions/AmazonCloudWatchAgent/ssm/amazon_linux.template
```

4. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
5. Choose **Create stack**.
6. For **Choose a template**, select **Upload a template to Amazon S3**, choose the template that you downloaded, and choose **Next**.
7. On the **Specify Details** page, fill out the following parameters accordingly and choose **Next**:
   - **Stack name**: Choose a stack name for your AWS CloudFormation stack.
   - **IAMRole**: Choose an IAM role that has permissions to write CloudWatch metrics and logs. For more information, see Creat IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances (p. 93).
   - **InstanceAMI**: Choose an AMI that is valid in the Region where you're going to launch your stack.
   - **InstanceType**: Choose a valid instance type.
   - **KeyName**: To enable SSH access to the new instance, choose an existing Amazon EC2 key pair. If you don't already have an Amazon EC2 key pair, you can create one in the AWS Management Console. For more information, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide for Linux Instances*.
   - **SSHLocation**: Specifies the IP address range that can be used to connect to the instance using SSH. The default allows access from any IP address.
   - **SSMKey**: Specifies the agent configuration file that you created and saved in Parameter Store.
8. On the **Options** page, you can choose to tag your stack resources. Choose **Next**.
9. On the **Review** page, review your information, acknowledge that the stack might create IAM resources, and then choose **Create**.

   If you refresh the console, you see that the new stack has the `CREATE_IN_PROGRESS` status.
10. When the instance is created, you can see it in the Amazon EC2 console. Optionally, you can connect to the host and check the progress.

    Use the following command to confirm that the agent is installed:

    ```
    rpm -qa amazon-cloudwatch-agent
    ```

    Use the following command to confirm that the agent is running:

    ```
    ps aux | grep amazon-cloudwatch-agent
    ```

The next procedure demonstrates using AWS CloudFormation to update the CloudWatch agent, using an agent configuration that you saved in Parameter Store.

To use AWS CloudFormation to update the CloudWatch agent using a configuration in Parameter Store

1. Change the agent configuration file stored in Parameter Store to the new configuration that you want.
2. In the AWS CloudFormation template that you downloaded in the the section called “Tutorial: Install the CloudWatch Agent Using AWS CloudFormation and Parameter Store” (p. 106) topic, change the version number. For example, you might change `VERSION=1.0` to `VERSION=2.0`.
3. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
4. On the AWS CloudFormation dashboard, select the stack that you created and choose **Update Stack**.
5. For **Select Template**, select **Upload a template to Amazon S3**, select the template that you just modified, and choose **Next**.
6. On the **Options** page, choose **Next** and then **Next**.

7. On the **Review** page, review your information and choose **Update**.

   After some time, you see `UPDATE_COMPLETE`.

Troubleshooting Installation of the CloudWatch Agent with AWS CloudFormation

This section helps you troubleshoot issues with installing and updating the CloudWatch agent using AWS CloudFormation.

Detecting When an Update Fails

If you use AWS CloudFormation to update your CloudWatch agent configuration, and use an invalid configuration, the agent stops sending any metrics to CloudWatch. A quick way to check whether an agent configuration update succeeded is to look at the `cfn-init-cmd.log` file. On a Linux server, the file is located at `/var/log/cfn-init-cmd.log`. On a Windows instance, the file is located at `C:\cfn\log\cfn-init-cmd.log`.

Metrics Are Missing

If you don't see metrics that you expect to see after installing or updating the agent, confirm that the agent is configured to collect that metric. To do this, check the `amazon-cloudwatch-agent.json` file to make sure that the metric is listed, and check that you are looking in the correct metric namespace. For more information, see CloudWatch Agent Files and Locations (p. 159).

Verifying the Signature of the CloudWatch Agent Package

GPG signature files are included for CloudWatch agent packages. You can use the public key to verify that the agent download file is original and unmodified. First, import the public key with https://gnupg.org/index.html.

To find the correct signature file, see the following table.

Architecture	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/	https://s3.amazonaws.com/amazoncloudwatch-agent/

Architecture	Platform	Download Link	Signature File Link
		suse/amd64/latest/amazon-cloudwatch-agent.rpm	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/amazon-cloudwatch-agent.msi	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig
arm64	Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
arm64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
arm64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig

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 doesn't match, don't install the agent. 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 the preceding table.

```
wget Signature File Link
```

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

```
shell$ gpg --verify signature-filename agent-download-filename
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 using the downloaded file.

Note the warning about trust. A key is trusted only if you or someone who you trust has signed it. This doesn't 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.

   You can perform the remaining steps 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 because 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 doesn't match, don't install the agent. 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. 86).
6. To verify the signature, run **gpg --verify**.

```
PS> gpg --verify sig-filename agent-download-filename
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 using the downloaded file.

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

Create the CloudWatch Agent Configuration File

Before running the CloudWatch agent on any servers, you must create a CloudWatch agent configuration file.

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 the file manually, the process is more complex, but you have more control over the metrics collected and can specify metrics not available through 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 save it manually as a JSON file and then use this file when installing the agent on your servers. Alternatively, you can store it in Systems Manager Parameter Store if you're going to use Systems Manager when you install the agent on servers.

Contents

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're 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*.

In the AWS credentials file, the wizard checks for default credentials and also looks for an `AmazonCloudWatchAgent` section such as the following:

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

The wizard displays the default credentials, the credentials from the `AmazonCloudWatchAgent`, and an `Others` option. You can select which credentials to use. If you choose `Others`, you can input credentials.

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 the CloudWatch Agent on On-Premises Servers (p. 94).

In the AWS configuration file, you can specify the Region that the agent sends metrics to if it's different than the `[default]` section. The default is to publish the metrics to the 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 predefined 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. 143).

Amazon EC2 instances running Linux

Detail Level	Metrics Included
Basic	**Mem:** mem_used_percent

Detail Level	Metrics Included
	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:** 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

Detail Level	Metrics Included
Advanced	**CPU:** `cpu_usage_guest`, `cpu_usage_idle`, `cpu_usage_iowait`, `cpu_usage_steal`, `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` **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	**Paging:** `Paging File % Usage` **Processor:** `Processor % Processor Time` **LogicalDisk:**`LogicalDisk % Free Space` **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`

Run the CloudWatch Agent Configuration Wizard

To create the CloudWatch agent configuration file

1. Start the CloudWatch agent configuration wizard by entering the following:

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
   ```

   On a server running Windows Server, enter 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're storing the configuration file locally, the configuration file `config.json` is stored in the current working directory. You can then copy this file to other servers where you want to install the agent.

   If you're 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 the CloudWatch Agent (p. 93).

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 doesn't create an `agent` section.
- The `metrics` section specifies the custom metrics for collection and publishing to CloudWatch. If you're 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 agent configuration 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. After you have created the file, you can copy it to other servers where you want to install the agent.

CloudWatch Agent Configuration File: Agent Section

The `agent` section can include the following fields. The wizard doesn't 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. You can override this value for specific types of metrics.

  This value 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. 44).

  The default value 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 isn't 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 Amazon Resource Name (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 Account (p. 155).
- `debug` – Optional. Specifies running the CloudWatch agent with debug log messages. The default value 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`

  The CloudWatch agent automatically rotates the log file that it creates. A log file is rotated out when it reaches 100 MB in size or is seven days old. The agent keeps as many as five backup log files that have been rotated out. Backup log files have a timestamp appended to their filename. The timestamp shows the date and time that the file was rotated out: for example, `amazon-cloudwatch-agent-2018-06-08T21-01-50.247.log.gz`.

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 value 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` – Optional. Specifies the dimensions that collected metrics are to be aggregated on. 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.

- `endpoint_override` – Specifies a FIPS endpoint or private link to use as the endpoint where the agent sends metrics. Specifying this and setting a private link enables you to send the metrics to an Amazon VPC endpoint. For more information, see What Is Amazon VPC?.

  The value of `endpoint_override` must be a string that is a URL.

- `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.

- `force_flush_interval` – Specifies in seconds the maximum amount of time that metrics remain in the memory buffer before being sent to the server. No matter the setting for this, if the size of the metrics in the buffer reaches 40 KB or 20 different metrics, the metrics are immediately sent to the server.

  The default value is 60.

- `credentials` – Specifies an IAM role to use when sending metrics to a different 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 account. For more information, see Sending Metrics and Logs to a Different Account (p. 155). If specified here, this value 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. 142).
- `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.
  - `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 don't have a unit.

Within the entry for each individual metric, you might 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 value 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. 44).
- `append_dimensions` – Optional. Additional dimensions to use for only the cpu metrics. If you specify this field, it's used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics that the agent collects.

- `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 value 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. 144).

    Within the entry for each individual metric, you might 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 value 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. 44).
  - `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 disk i/o 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`.

    Within the entry for each individual metric, you might 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 diskio metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This value 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. 44).
  - `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 as many as three fields:
  - `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. 144).

    Within the entry for each individual metric, you might 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 swap metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This value 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. 44).
  - `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's 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 as many as three fields:

- `measurement` – Specifies the array of memory 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. 144).

  Within the entry for each individual metric, you might 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 value 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. 44).
- `append_dimensions` – Optional. Additional dimensions to use for only the mem metrics. If you specify this field, it's used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics that the agent collects.

- `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 four 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. 144).

    Within the entry for each individual metric, you might 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 net metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This value 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. 44).
  - `append_dimensions` – Optional. Additional dimensions to use for only the net metrics. If you specify this field, it's 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 as many as three fields:
  - `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. 144).

    Within the entry for each individual metric, you might 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 value 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. 44).
  - `append_dimensions` – Optional. Additional dimensions to use for only the netstat metrics. If you specify this field, it's 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 as many as three fields:
  - `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 might 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 processes metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

    This value 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. 44).
  - `append_dimensions` – Optional. Additional dimensions to use for only the process metrics. If you specify this field, it's used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.
- `procstat` – Optional. Specifies that you want to retrieve metrics from individual processes. For more information, see Collect Process Metrics with the procstat Plugin (p. 134).

- `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. 141).

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 to collect metrics from. 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. For objects that don't have instances, omit `resources`.

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 value 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. 44).
- `append_dimensions` – Optional. Specifies additional dimensions to use for only the metrics for this object. If you specify this field, it's 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 that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in MetricDatum.

There are two other optional sections that you can include in `metrics_collected`:

- `statsd` – Enables you 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. 141).
- `procstat` – Enables you to retrieve metrics from individual processes. For more information, see Collect Process Metrics with the procstat Plugin (p. 134).

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 the CloudWatch agent is to collect. 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.

      You can also use the standard asterisk 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 that 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, and `{local_hostname}` uses the hostname from the network configuration file.

      If you omit this field, the default value of `{instance_id}` is used. If a log stream doesn't already exist, it's created automatically.
    - `timezone` – Optional. Specifies the time zone to use when putting timestamps on log events. The valid values are `UTC` and `Local`. The default value is `Local`.
    - `timestamp_format` – Optional. Specifies the timestamp 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 symbols in the following list as part of the format. This list of symbols is different than the list used by the older CloudWatch Logs agent. For a summary of these differences, see Timestamp Differences Between the Unified CloudWatch Agent and the Older CloudWatch Logs Agent (p. 156)

      `%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` isn't 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 subsequent 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 timestamp 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 can't be decoded are replaced with other characters.

  The default value is `utf-8`. The following 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. This is equivalent to the Windows event log channel name: 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, and `{local_hostname}` uses the hostname from the network configuration file.

      If you omit this field, the default value of `{instance_id}` is used. If a log stream doesn't already exist, it's 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 don't have individual log stream names defined in their entry in `collect_list`.
- `endpoint_override` – Specifies a FIPS endpoint or private link to use as the endpoint where the agent sends logs. Specifying this field and setting a private link enables you to send the logs to an Amazon VPC endpoint. For more information, see What Is Amazon VPC?.

  The value of `endpoint_override` must be a string that is a URL.
- `force_flush_interval` – Specifies in seconds the maximum amount of time that logs remain in the memory buffer before being sent to the server. No matter the setting for this field, if the size of the logs in the buffer reaches 1 MB, the logs are immediately sent to the server. The default value is 5.
- `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

Account (p. 155). 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": "CustomizedName",
                       "event_levels": [
                           "INFORMATION",
                           "ERROR"
                       ],
                       "log_group_name": "CustomizedLogGroup",
                       "log_stream_name": "CustomizedLogStream"
                   }
               ]
           }
       },
       "log_stream_name": "my_log_stream_name"
}
```

CloudWatch Agent Configuration File: Complete Examples

The following is an example of a complete CloudWatch 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"],
[]],
        "force_flush_interval" : 30
      },
      "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",
        "force_flush_interval" : 15
      }
    }
```

The following is an example of a complete CloudWatch 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": "CustomizedName",
                "event_levels": [
                  "WARNING",
                  "ERROR"
                ],
                "log_group_name": "CustomizedLogGroup",
                "log_stream_name": "CustomizedLogStream",
                "event_format": "xml"
              }
            ]
          }
        },
        "log_stream_name": "example_log_stream_name"
      }
    }
```

Save the CloudWatch Agent Configuration File Manually

If you create or edit the CloudWatch agent configuration 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. After you have created the file, you can copy it to other servers where you want to run the agent.

Uploading the CloudWatch Agent Configuration File to Systems Manager Parameter Store

If you plan to use the SSM Agent to install the CloudWatch agent on servers, after you manually edit the CloudWatch agent configuration file, you can upload it to Systems Manager Parameter Store. To do so, 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 the CloudWatch Agent (p. 93).

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 file name of the configuration file that you edited.

```
aws ssm put-parameter --name "parameter name" --type "String" --value
 file://configuration_file_pathname
```

Collect Process Metrics with the procstat Plugin

The *procstat* plugin enables you to collect metrics from individual processes. It is supported on Linux servers and on servers running Windows Server 2008 or later.

Topics

Configuring the CloudWatch Agent for procstat

To use the procstat plugin, add a `procstat` section in the `metrics_collected` section of the CloudWatch agent configuration file. There are three ways to specify the processes to monitor. You can use only one of these methods, but you can use that method to specify one or more processes to monitor.

- `pid_file`: Selects processes by the names of the process identification number (PID) files they create.
- `exe`: Selects the processes that have process names that match the string that you specify, using regular expression matching rules. For more information, see Syntax.
- `pattern`: Selects processes by the command lines used to start the processes. All processes are selected that have command lines matching the specified string using regular expression matching rules. The entire command line is checked, including parameters and options used with the command.

The CloudWatch agent uses only one of these methods, even if you include more than one of the above sections. If you specify more than one section, the CloudWatch agent uses the `pid_file` section if it is present. If not, it uses the `exe` section.

On Linux servers, the strings that you specify in an `exe` or `pattern` section are evaluated as regular expressions. On servers running Windows Server, these strings are evaluated as WMI queries. For more information, see LIKE Operator.

Whichever method you use, you can include an optional `metrics_collection_interval` parameter, which specifies how often in seconds to collect those metrics. If you omit this parameter, the default value of 60 seconds is used.

In the examples in the following sections, the `procstat` section is the only section included in the `metrics_collected` section of the agent configuration file. Actual configuration files can also include other sections in `metrics_collected`. For more information, see Manually Create or Edit the CloudWatch Agent Configuration File (p. 116).

Configuring with Pid_file

The following example `procstat` section monitors the processes that create the PID files `example1.pid` and `example2.pid`. Different metrics are collected from each process. Metrics collected from the process that creates `example2.pid` are collected every 10 seconds, and the metrics collected from the `example1.pid` process are collected every 60 seconds, the default value.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pid_file": "/var/run/example1.pid",
                    "measurement": [
                        "cpu_usage",
                        "memory_rss"
                    ]
                },
                {
                    "pid_file": "/var/run/example2.pid",
                    "measurement": [
                        "read_bytes",
                        "read_count",
                        "write_bytes"
                    ],
                    "metrics_collection_interval": 10
                }
            ]
        }
    }
}
```

Configuring with Exe

The following example `procstat` section monitors all processes with names that match the strings `agent` or `plugin`. The same metrics are collected from each process.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "exe": "agent",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                },
                {
                    "exe": "plugin",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
```

```
                    "cpu_time_user"
                ]
            }
        ]
    }
}
```

Configuring with Pattern

The following example `procstat` section monitors all processes with command lines that match the strings `config` or `-c`. The same metrics are collected from each process.

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pattern": "config",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                },
                {
                    "pattern": "-c",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                }
            ]
        }
    }
}
```

Metrics Collected by Procstat

The following table lists the metrics that you can collect with the `procstat` plugin.

The CloudWatch agent adds `procstat` to the beginning of the following metric names. There is a different syntax depending on whether it was collected from a Linux server or a server running Windows Server. For example, the `cpu_time` metric appears as `procstat_cpu_time` when collected from Linux and as `procstat cpu_time` when collected from Windows Server.

Metric Name	Available On	Description
`cpu_time`	Linux	The amount of time that the process uses the CPU. This metric is measured in hundredths of a second. Unit: Count

Metric Name	Available On	Description
cpu_time_system	Linux, Windows Server	The amount of time that the process is in system mode. This metric is measured in hundredths of a second. Type: Float Unit: Count
cpu_time_user	Linux, Windows Server	The amount of time that the process is in user mode. This metric is measured in hundredths of a second. Unit: Count
cpu_usage	Linux, Windows Server	The percentage of time that the process is active in any capacity. Unit: Percent
memory_data	Linux	The amount of memory that the process uses for data. Unit: Bytes
memory_locked	Linux	The amount of memory that the process has locked. Unit: Bytes
memory_rss	Linux, Windows Server	The amount of real memory (resident set) that the process is using. Unit: Bytes

Metric Name	Available On	Description
memory_stack	Linux	The amount of stack memory that the process is using. Unit: Bytes
memory_swap	Linux	The amount of swap memory that the process is using. Unit: Bytes
memory_vms	Linux, Windows Server	The amount of virtual memory that the process is using. Unit: Bytes
read_bytes	Linux, Windows Server	The number of bytes that the process has read from disks. Unit: Bytes
write_bytes	Linux, Windows Server	The number of bytes that the process has written to disks. Unit: Bytes
read_count	Linux, Windows Server	The number of disk read operations that the process has executed. Unit: Count
write_count	Linux, Windows Server	The number of disk write operations that the process has executed. Unit: Count

Metric Name	Available On	Description
involuntary_context_switches	Linux	The number of times that the process was involuntarily context-switched. Unit: Count
voluntary_context_switches	Linux	The number of times that the process was context-switched voluntarily. Unit: Count
realtime_priority	Linux	The current usage of real-time priority for the process. Unit: Count
nice_priority	Linux	The current usage of nice priority for the process. Unit: Count
signals_pending	Linux	The number of signals pending to be handled by the process. Unit: Count
rlimit_cpu_time_hard	Linux	The hard CPU time resource limit for the process. Unit: Count
rlimit_cpu_time_soft	Linux	The soft CPU time resource limit for the process. Unit: Count

Metric Name	Available On	Description
rlimit_file_locks_hard	Linux	The hard file locks resource limit for the process. Unit: Count
rlimit_file_locks_soft	Linux	The soft file locks resource limit for the process. Unit: Count
rlimit_memory_data_hard	Linux	The hard resource limit on the process for memory used for data. Unit: Bytes
rlimit_memory_data_soft	Linux	The soft resource limit on the process for memory used for data. Unit: Bytes
rlimit_memory_locked_hard	Linux	The hard resource limit on the process for locked memory. Unit: Bytes
rlimit_memory_locked_soft	Linux	The soft resource limit on the process for locked memory. Unit: Bytes
rlimit_memory_rss_hard	Linux	The hard resource limit on the process for physical memory. Unit: Bytes

Metric Name	Available On	Description
rlimit_memory_rss_soft	Linux	The soft resource limit on the process for physical memory. Unit: Bytes
rlimit_memory_stack_hard	Linux	The hard resource limit on the process stack. Unit: Bytes
rlimit_memory_stack_soft	Linux	The soft resource limit on the process stack. Unit: Bytes
rlimit_memory_vms_hard	Linux	The hard resource limit on the process for virtual memory. Unit: Bytes

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 value 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.

To collect these custom metrics, 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's done for you. For more information, see Create the CloudWatch Agent Configuration File (p. 111).

The `StatsD` default configuration works for most users. There are three optional fields that 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 agent listens on all available interfaces. Only the UDP format is supported, so you don't need to specify a UDP prefix.

  The default value is `:8125`.
- `metrics_collection_interval` – How often in seconds that the `StatsD` plugin runs and collects metrics. The default value is 10 seconds. The range is 1–172,000.
- `metrics_aggregation_interval` – How often in seconds CloudWatch aggregates metrics into single data points. The default value 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 0–172,000. 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, which is supported only on Linux servers. You use the collectd software to send the metrics to the CloudWatch agent. For the collectd metrics, the CloudWatch agent acts as the server while the collectd plug-in acts as the client.

The collectd software is not installed automatically on every server. For more information, see the Download page for collectd.

To collect these custom metrics, 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. 111).

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_typesdb:** 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
         }
      }
   }
}
```

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.

For information about installing the CloudWatch agent on an instance, see Collecting Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent (p. 84).

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 following table lists metrics that you can collect with the CloudWatch agent on Linux instances.

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

Metric	Description
	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
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

Metric	Description
cpu_usage_nice	The percentage of time that the CPU is in user mode with low-priority processes, which higher-priority processes can easily interrupt. Unit: Percent
cpu_usage_softirq	The percentage of time that the CPU is servicing software interrupts. Unit: Percent
cpu_usage_steal	The percentage of time that the CPU is in *stolen time*, or 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

Metric	Description
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
diskio_read_time	The amount of time that read requests have waited on the disks. Multiple read requests waiting at the same time increase the number. For example, if 5 requests all wait for an average of 100 milliseconds, 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 increase the number. For example, if 8 requests all wait for an average of 1000 milliseconds, 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

Metric	Description
mem_free	The amount of memory that isn't being used. Unit: Bytes
mem_inactive	The amount of memory that hasn't been used in some way during the last sample period Unit: Bytes
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 that were dropped. Unit: Count
net_drop_out	The number of packets transmitted by this network interface that 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

Metric	Description
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
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

Metric	Description
netstat_tcp_time_wait	The number of TCP connections currently waiting to ensure that the client received the acknowledgement of its connection termination request. Unit: Count
netstat_udp_socket	The number of current UDP connections. Unit: Count
processes_blocked	The number of processes that are blocked. Unit: Count
processes_dead	The number of processes that are dead, 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, 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, indicated by the `W` state code on FreeBSD instances. This metric is available only on FreeBSD instances. Unit: Count

Metric	Description
processes_zombies	The number of zombie processes, indicated by the Z state code. Unit: Count
swap_free	The amount of swap space that isn't being used. Unit: Bytes
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

Common Scenarios with the 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 restart the agent to have the changes take effect.

Multiple CloudWatch Agent Configuration Files

You can set up the CloudWatch agent to use multiple configuration files. For example, you can use a common configuration file that collects a set of metrics and logs that you always want to collect from all servers in your infrastructure. You can then use additional configuration files that collect metrics from certain applications or in certain situations.

To set this up, first create the configuration files that you want to use. Any configuration files that will be used together on the same server must have different file names. You can store the configuration files on servers or in Parameter Store.

Start the CloudWatch agent using the `fetch-config` option and specify the first configuration file. To append the second configuration file to the running agent, use the same command but with the `append-config` option. All metrics and logs listed in either configuration file are collected. The following example Linux commands illustrate this scenario using configurations stores as files. The first line starts the agent using the `infrastructure.json` configuration file, and the second line appends the `app.json` configuration file.

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -
c file:/tmp/infrastructure.json -s
```

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a append-config -m ec2 -
c file:/tmp/app.json -s
```

The following example configuration files illustrate a use for this feature. The first configuration file is used for all servers in the infrastructure, and the second collects only logs from a certain application and is appended to servers running that application.

infrastructure.json

```
{
  "metrics": {
    "metrics_collected": {
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          "usage_active"
```

```
        ],
        "totalcpu": true
      },
      "mem": {
         "measurement": [
           "used_percent"
        ]
      }
}
  },
  "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"
          },
          {
            "file_path": "/var/log/messages",
            "log_group_name": "/var/log/messages"
          }
        ]
      }
    }
  }
}
```

app.json

```
{
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/app/app.log*",
            "log_group_name": "/app/app.log"
          }
        ]
      }
    }
  }
```

Any configuration files appended to the configuration must have different file names from each other and from the initial configuration file. If you use `append-config` with a configuration file with the same file name as a configuration file that the agent is already using, the append command overwrites the information from the first configuration file instead of appending to it. This is true even if the two configuration files with the same file name are on different file paths.

The preceding example shows the use of two configuration files, but there is no limit to the number of configuration files that you can append to the agent configuration. You can also mix the use of configuration files located on servers and configurations located in Parameter Store.

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 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, and 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 restart the agent to have the changes take effect.

Sending Metrics and Logs to a Different Account

To have the CloudWatch agent send the metrics, logs, or both to a different 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 target 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 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 Creat IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances (p. 93).

The role in the sending account also must include the following policy. You add this policy on 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 on 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 Creat IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances (p. 93). 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"
    }
  ]
}
```

Timestamp Differences Between the Unified CloudWatch Agent and the Older CloudWatch Logs Agent

The CloudWatch agent supports a different set of symbols for timestamp formats, compared to the older CloudWatch Logs agent. These differences are shown in the following table.

Symbols Supported by Both Agents	Symbols Supported Only by Unified CloudWatch Agent	Symbols Supported Only by Older CloudWatch Logs Agent
%A, %a, %b, %B, %d, %H, %l, %m, %M, %p, %S, %y, %Y, %Z, %z	%-d, %-l, %-m, %-M, %-S	%c, %f, %j, %U, %W, %w

For more information about the meanings of the symbols supported by the new CloudWatch agent, see CloudWatch Agent Configuration File: Logs Section in the *Amazon CloudWatch User Guide*. For information about symbols supported by the CloudWatch Logs agent, see Agent Configuration File in the *Amazon CloudWatch Logs User Guide*.

Troubleshooting the CloudWatch Agent

Use the following information to help troubleshoot problems with the CloudWatch agent.

Topics

CloudWatch Agent Command Line Parameters

To see the full list of parameters supported by the CloudWatch agent, enter the following at the command line at a computer where you have it installed:

```
amazon-cloudwatch-agent-ctl -help
```

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 isn't the correct version, you might 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 in the *AWS Systems Manager User Guide*.

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's 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. Keep **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, enter the following:

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status
  ```

  On a server running Windows Server, enter 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 can't find metrics collected by it in the AWS Management Console or the AWS CLI, confirm that you're 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 don't 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 in the directory where you ran the configuration wizard, or you might 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. 117).

The CloudWatch Agent Won't Start, and the Error Mentions an Amazon EC2 Region

If the agent doesn't start and the error message mentions an Amazon EC2 Region endpoint, you might 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*.

Unable to Find Credentials on Windows Server

On Windows Server, if you have credentials in a location other than `$SystemDrive\\Users\\Administrator\\.aws` on Windows Server 2008 or Windows Server 2012, or `"$SystemDrive\Documents and Settings\\Administrator\\.aws` on Windows Server 2003, you can specify your own credential path by using the `shared_credential_file` option in `common.toml`.

If you don't have a credential file, you must create one. For more information, see (Optional) Modify the Common Configuration for Proxy or Region Information (p. 91).

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` or `/usr/bin/amazon-cloudwatch-agent-ctl`	`$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1`
The log file the agent writes to. You might need to attach this when contacting AWS Support.	`/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` or `/var/log/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.log`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log`

File	Linux Location	Windows Server Location
Agent configuration validation file.	`/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` or `/var/log/amazon/amazon-cloudwatch-agent/configuration-validation.log`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log`
The JSON file used to configure the agent immediately after the wizard creates it. For more information, see Create the CloudWatch Agent Configuration File (p. 111).	`/opt/aws/amazon-cloudwatch-agent/bin/config.json`	`$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` or `/etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`
TOML file used to specify Region and credential information to be used by the agent, overriding system defaults.	`/opt/aws/amazon-cloudwatch-agent/etc/common-config.toml` or `/etc/amazon/amazon-cloudwatch-agent/common-config.toml`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.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. 116).

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. Keep **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, enter the following:

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a stop
```

On a server running Windows Server, enter 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 Start the CloudWatch Agent (p. 99).

AWS Services That Publish CloudWatch Metrics

The following AWS services publish metrics to CloudWatch. For information about the metrics and dimensions, see the specified documentation.

Service	Namespace	Documentation
Amazon API Gateway	`AWS/ApiGateway`	Monitor API Execution with Amazon CloudWatch
AppStream 2.0	`AWS/AppStream`	Monitoring Amazon AppStream 2.0 Resources
Amazon Athena	`AWS/Athena`	Monitoring Athena Queries with CloudWatch Metrics
AWS Billing and Cost Management	`AWS/Billing`	Monitoring Charges with Alerts and Notifications
ACM Private CA	`AWS/ACMPrivateCA`	Supported CloudWatch Metrics
Amazon CloudFront	`AWS/CloudFront`	Monitoring CloudFront Activity Using CloudWatch
Amazon CloudSearch	`AWS/CloudSearch`	Monitoring an Amazon CloudSearch Domain with Amazon CloudWatch
Amazon CloudWatch Events	`AWS/Events`	Monitoring Usage with CloudWatch Metrics
Amazon CloudWatch Logs	`AWS/Logs`	Monitoring Usage with CloudWatch Metrics
AWS CodeBuild	`AWS/CodeBuild`	Monitoring AWS CodeBuild
Amazon Cognito	`AWS/Cognito`	Viewing Advanced Security Metrics
Amazon Connect	`AWS/Connect`	Monitoring Amazon Connect in Amazon CloudWatch Metrics
AWS Database Migration Service	`AWS/DMS`	Monitoring AWS DMS Tasks
AWS Direct Connect	`AWS/DX`	Monitoring with Amazon CloudWatch
Amazon DynamoDB	`AWS/DynamoDB`	Monitoring DynamoDB
Amazon EC2	`AWS/EC2`	Monitoring Your Instances Using CloudWatch
Amazon EC2 Spot Fleet	`AWS/EC2Spot`	CloudWatch Metrics for Spot Fleet
Amazon EC2 Auto Scaling	`AWS/AutoScaling`	Monitoring Your Auto Scaling Groups and Instances Using CloudWatch

Service	Namespace	Documentation
AWS Elastic Beanstalk	`AWS/ElasticBeanstalk`	Publishing Amazon CloudWatch Custom Metrics for an Environment
Amazon Elastic Block Store	`AWS/EBS`	Monitoring the Status of Your Volumes
Amazon Elastic Container Service	`AWS/ECS`	Amazon ECS CloudWatch Metrics
Amazon Elastic File System	`AWS/EFS`	Monitoring with CloudWatch
Amazon Elastic Inference	`AWS/ElasticInference`	Using CloudWatch Metrics to Monitor Amazon EI
Elastic Load Balancing	`AWS/ApplicationELB`	CloudWatch Metrics for Your Application Load Balancer
Elastic Load Balancing	`AWS/ELB`	CloudWatch Metrics for Your Classic Load Balancer
Elastic Load Balancing	`AWS/NetworkELB`	CloudWatch Metrics for Your Network Load Balancer
Amazon Elastic Transcoder	`AWS/ElasticTranscoder`	Monitoring with Amazon CloudWatch
Amazon ElastiCache for Memcached	`AWS/ElastiCache`	Monitoring Use with CloudWatch Metrics
Amazon ElastiCache for Redis	`AWS/ElastiCache`	Monitoring Use with CloudWatch Metrics
Amazon Elasticsearch Service	`AWS/ES`	Monitoring Cluster Metrics and Statistics with CloudWatch
Amazon EMR	`AWS/ElasticMapReduce`	Monitor Metrics with CloudWatch
AWS Elemental MediaConnect	`AWS/MediaConnect`	Monitoring AWS Elemental MediaConnect with Amazon CloudWatch
AWS Elemental MediaConvert	`AWS/MediaConvert`	CloudWatch Metrics
AWS Elemental MediaPackage	`AWS/MediaPackage`	CloudWatch Metrics
AWS Elemental MediaTailor	`AWS/MediaTailor`	Monitoring AWS Elemental MediaTailor with Amazon CloudWatch
Amazon FSx for Lustre	`AWS/FSx`	Monitoring Amazon FSx for Lustre
Amazon GameLift	`AWS/GameLift`	Monitor Amazon GameLift with CloudWatch

Service	Namespace	Documentation
AWS Glue	`AWS/Glue`	Monitoring AWS Glue Using CloudWatch Metrics
Amazon Inspector	`AWS/Inspector`	Monitoring Amazon Inspector Using CloudWatch
AWS IoT	`AWS/IoT`	Monitoring with Amazon CloudWatch
AWS IoT Analytics	`AWS/IoTAnalytics`	Namespace, Metrics, and Dimensions
AWS IoT Things Graph	`AWS/ThingsGraph`	Metrics
AWS Key Management Service	`AWS/KMS`	Monitoring with CloudWatch
Amazon Kinesis Data Analytics	`AWS/KinesisAnalytics`	*Kinesis Data Analytics*: Monitoring with CloudWatch *Kinesis Data Analytics for Java Applications*: Viewing Amazon Kinesis Data Analytics Metrics and Dimensions
Amazon Kinesis Data Firehose	`AWS/Firehose`	Monitoring Kinesis Data Firehose Using CloudWatch Metrics
Amazon Kinesis Data Streams	`AWS/Kinesis`	Monitoring Amazon Kinesis Data Streams with Amazon CloudWatch
Amazon Kinesis Video Streams	`AWS/KinesisVideo`	Monitoring Kinesis Video Streams Metrics with CloudWatch
AWS Lambda	`AWS/Lambda`	AWS Lambda Metrics
Amazon Lex	`AWS/Lex`	Monitoring Amazon Lex with CloudWatch
Amazon Machine Learning	`AWS/ML`	Monitoring Amazon ML with CloudWatch Metrics
Amazon Managed Streaming for Kafka	`AWS/Kafka`	Monitoring Amazon MSK with Amazon CloudWatch
Amazon MQ	`AWS/AmazonMQ`	Monitoring Amazon MQ Brokers Using Amazon CloudWatch
Amazon Neptune	`AWS/Neptune`	Monitoring Neptune with CloudWatch
AWS OpsWorks	`AWS/OpsWorks`	Monitoring Stacks using Amazon CloudWatch
Amazon Polly	`AWS/Polly`	Integrating CloudWatch with Amazon Polly
Amazon Redshift	`AWS/Redshift`	Amazon Redshift Performance Data
Amazon Relational Database Service	`AWS/RDS`	Monitoring with Amazon CloudWatch
Amazon Route 53	`AWS/Route53`	Monitoring Amazon Route 53
Amazon SageMaker	`AWS/SageMaker`	Monitoring Amazon SageMaker with CloudWatch

Service	Namespace	Documentation
AWS Shield Advanced	`AWS/DDoSProtection`	Monitoring with CloudWatch
Amazon Simple Email Service	`AWS/SES`	Retrieving Amazon SES Event Data from CloudWatch
Amazon Simple Notification Service	`AWS/SNS`	Monitoring Amazon SNS with CloudWatch
Amazon Simple Queue Service	`AWS/SQS`	Monitoring Amazon SQS Queues Using CloudWatch
Amazon Simple Storage Service	`AWS/S3`	Monitoring Metrics with Amazon CloudWatch
Amazon Simple Workflow Service	`AWS/SWF`	Amazon SWF Metrics for CloudWatch
AWS Step Functions	`AWS/States`	Monitoring Step Functions Using CloudWatch
AWS Storage Gateway	`AWS/StorageGateway`	Monitoring Your Gateway and Resources
Amazon Textract	`AWS/Textract`	CloudWatch Metrics for Amazon Textract
Amazon Translate	`AWS/Translate`	CloudWatch Metrics and Dimensions for Amazon Translate
AWS Trusted Advisor	`AWS/TrustedAdvisor`	Creating Trusted Advisor Alarms Using CloudWatch
Amazon VPC	`AWS/NATGateway`	Monitoring Your NAT Gateway with CloudWatch
Amazon VPC	`AWS/TransitGateway`	CloudWatch Metrics for Your Transit Gateways
Amazon VPC	`AWS/VPN`	Monitoring with CloudWatch
AWS WAF	`WAF`	Monitoring with CloudWatch
Amazon WorkSpaces	`AWS/WorkSpaces`	Monitor Your WorkSpaces Using CloudWatch Metrics

Monitor Applications Using AWS SDK Metrics

Enterprise customers can use the CloudWatch agent with AWS SDK Metrics for Enterprise Support (SDK Metrics) to collect metrics from AWS SDKs on their hosts and clients. These metrics are shared with AWS Enterprise Support. SDK Metrics can help you collect relevant metrics and diagnostic data about your application's connections to AWS services without adding custom instrumentation to your code, and reduces the manual work necessary to share logs and data with AWS Support.

> **Important**
> SDK Metrics is available only to customers with an Enterprise Support subscription. For more information, see Amazon CloudWatch Support Center.

You can use SDK Metrics with any application that directly calls AWS services and that was built using an AWS SDK that is one of the versions listed in the following section.

SDK Metrics instruments calls made by the AWS SDK and uses the CloudWatch agent running in the same environment as a client application that is using an AWS SDK. The following section explains the steps necessary to enable the CloudWatch agent to emit SDK Metrics data. For information about what you need to configure in your SDK, see your SDK documentation.

Topics

Supported Versions

To use SDK Metrics, you must be using version 1.207573.0 or later of the CloudWatch agent. If you are already running the CloudWatch agent, you can find the version in the following file.

- Linux: `/opt/aws/amazon-cloudwatch-agent/bin/CWAGENT_VERSION`
- Windows Server: `C:\Program Files\Amazon\AmazonCloudWatchAgent\CWAGENT_VERSION`

Additionally, you must be using one of the following versions of an AWS SDK.

- AWS CLI 1.16.84 or later
- AWS SDK for C++ 1.7.47 or later
- AWS SDK for Go 1.16.18 or later
- AWS SDK for Java 1.11.482 or later
- AWS SDK for JavaScript in Node.js 2.387 or later
- AWS SDK for .NET 3.3.440 or later
- AWS SDK for PHP 3.85.0 or later
- AWS SDK for Python (Boto 3) 1.9.78 or later
- AWS SDK for Ruby 3.45.0 or later

Metrics and Data Collected by AWS SDK Metrics for Enterprise Support

SDK Metrics collects data from your applications and uses it to emit metrics to CloudWatch. The following table lists the data collected by SDK Metrics.

Data	Type
Message Version	String
Message ID	String
Service Endpoint	String
Normalized Service ID	String
API Operation name	String
Availability (from SDK customer point of view)	Integer (0 or 1) with number of samples
Latency (from SDK customer point of view)	Distribution
SDK Version	String
Client Language Runtime Version	String
Client Operating System	String
Service Response Codes	Key/value pairs
Client Language Runtime Version	String
Sample Request IDs	List
Retries	Distribution
Throttled Requests	Distribution
AccountID	String
Availability Zone	String
Instance ID	String
Runtime Environment (Lambda/ECS)	String
Network Error Messages	String/map
Source IP Address	String
Destination IP Address	String

The following table lists the metrics that Enterprise Support customers can collect using AWS SDK Metrics for Enterprise Support. These metrics are in the `AWS/SDKMetrics` namespace.

AWS Support resources and your technical account manager should have access to SDK Metrics data to help you resolve cases. If you discover data that is confusing or unexpected but doesn't negatively

impact your application performance, we recommend that you wait and review that data during scheduled business reviews with your technical account manager.

Metric	Description
CallCount	Total number of successful or failed API calls from your code to AWS services. Use `CallCount` as a baseline to correlate with other metrics such as `ServerErrorCount` and `ThrottleCount`. Unit: Count
ClientErrorCount	The number of API calls that failed with client errors (4xx HTTP response codes). These can include throttling errors, *access denied*, *S3 bucket does not exist*, and *invalid parameter value*. A high value for this metric usually indicates that something in your application needs to be fixed, unless the high value is a result of throttling caused by an AWS service limit. In this case, you should increase your service limit. Unit: Count
EndToEndLatency	The total time for your application to make a call using the AWS SDK, inclusive of retries. Use `EndToEndLatency` to determine how AWS API calls contribute to your application's overall latency. Higher than expected latency might be caused by issues with your network, firewall, or other configuration settings. Latency can also result from SDK retries. Unit: Milliseconds
ConnectionErrorCount	The number of API calls that fail because of errors connecting to the service. These can be caused by network issues between your application and AWS services, including load balancer issues, DNS failures, and transit provider issues. In some cases, AWS issues might result in this error. Use this metric to determine whether problems are specific to your application or are caused by your infrastructure or network. A high value could also indicate short timeout values for API calls. Unit: Count
ServerErrorCount	The number of API calls that fail because of server errors (5xx HTTP response codes) from AWS services. These are typically caused by AWS services. Use this metric to determine the cause of SDK retries or latency. This metric doesn't always indicate that AWS services are at fault because some AWS teams classify latency as an HTTP 503 response. Unit: Count

Metric	Description
ThrottleCount	The number of API calls that fail because of throttling by AWS services. Use this metric to assess if your application has reached throttle limits, as well as to determine the cause of retries and application latency. If you see high values, consider distributing calls over a window instead of batching your calls. Unit: Count

You can use the following dimensions with SDK Metrics.

Dimension	Description
DestinationRegion	The AWS Region that is the destination of the call.
Service	The AWS service being called by the application.

Configure the CloudWatch Agent for SDK Metrics

Before you begin, install the CloudWatch agent on an EC2 instance running an application for which you'd like to receive metrics. For more information, see Installing the CloudWatch Agent (p. 85).

After installing the CloudWatch agent, you need to configure it to work with SDK Metrics. The easiest way is to use AWS Systems Manager, but you can also do it manually.

Topics

Configure the CloudWatch Agent for SDK Metrics Using AWS Systems Manager

This section explains how to use SSM to configure the CloudWatch agent to work with SDK Metrics. For more information about SSM agents, see Installing and Configuring SSM Agent.

To configure SDK Metrics using SSM

1. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/.
2. In the navigation pane, choose **Run Command**.
3. In the **Command document** list, choose **AWS-UpdateSSMAgent**.
4. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
5. In the navigation pane, choose **Parameter Store**.
6. Choose **Create Parameter**.
7. Do the following:

    a. Name the parameter `AmazonCSM`.

b. Select type *string*.
c. For **Value**, enter `{ "CSM": {"memory_limit_in_mb": 20, "port": 31000}}`.
8. Select **Create Parameter**.

To complete the configuration, see your SDK documentation.

Configure the CloudWatch Agent for SDK Metrics Manually

This section explains how to manually configure the CloudWatch agent to work with SDK Metrics.

Create Your CloudWatch Agent Configuration File

If you don't already have a CloudWatch agent configuration file, create one. For more information, see Create the CloudWatch Agent Configuration File (p. 111).

Add the SDK Metrics Configuration

Add the following `csm` entry to the top-level JSON object in the CloudWatch agent configuration file.

```
"csm": {
    "memory_limit_in_mb": 20,
    "port": 31000
  }
```

For example:

```
{
    "agent": {
      ...
    },
    "metrics": {
      ...
    },
    "csm": {
      "memory_limit_in_mb": 20,
      "port": 31000
    }
  }
```

You can now start the CloudWatch agent using this configuration file. For more information, see Start the CloudWatch Agent (p. 99).

To complete the configuration, see your SDK documentation.

Set IAM Permissions for SDK Metrics

To configure permissions so you can use SDK Metrics, you must create an IAM policy to allow the SDK Metrics process and attach it to the role or user managing the EC2 instance.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy**, **JSON**.

4. Enter the following inline policy into the content window. Ignore the warning about the non-supported action.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sdkmetrics:*"
            ],
            "Resource": "*"
        }
    ]
}
```

5. Name the policy **AmazonSDKMetrics**.
6. Attach the policy to the IAM user or role that manages the EC2 instances that you want to monitor.

CloudWatch Tutorials

The following scenarios illustrate uses of 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 (AWS 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 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.

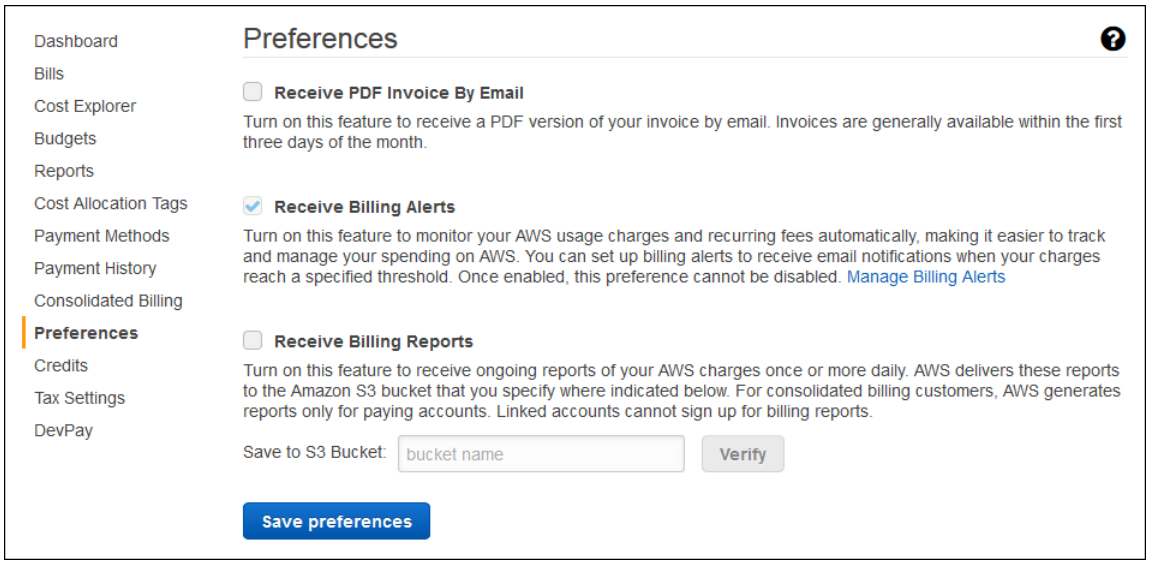
Requirements

- 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 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. 82) 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**, **Create Alarm**.
4. Choose **Select metric**, **Billing**, **Total Estimated Charge**.
5. Select the checkbox next to **EstimatedCharges** and choose **Select metric**
6. 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**
   > The graph shows a current estimate of your charges that you can use to set an appropriate amount.
7. 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 is sent a subscription confirmation email. The recipient must confirm the subscription before notifications can be sent to the email address.

8. 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**.
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.

Step 4: Edit a Billing Alarm

For example, you may want to increase the amount of 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**.
4. Select the check box next to the alarm and 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

If you no longer need your billing alarm, you can delete 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**.
4. Select the check box next to the alarm and choose **Actions**, **Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Scenario: Publish Metrics to CloudWatch

In this scenario, you 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

In this scenario, you 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 publish several data points that collectively represent three hours of latency data. The raw data comprises 15 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 publish the data points from hour one as single data points. For the data from hours two and three, you 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 timestamp with a timestamp 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 timestamp 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 timestamp 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 timestamp 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. On 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 Graphing Metrics (p. 39).

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. The endpoint name will be `com.amazonaws.Region.monitoring`. 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

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. 182)
- Using Identity-Based Policies (IAM Policies) for CloudWatch (p. 185)
- Amazon CloudWatch Permissions Reference (p. 194)

CloudWatch Dashboard Permissions Update

On May 1, 2018, AWS changed the permissions required to access CloudWatch dashboards. Dashboard access in the CloudWatch console now requires permissions that were introduced in 2017 to support dashboard API operations:

- **cloudwatch:GetDashboard**
- **cloudwatch:ListDashboards**
- **cloudwatch:PutDashboard**
- **cloudwatch:DeleteDashboards**

To access 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. 194).

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. 185).

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)

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. 185). 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. 182) 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. 182).

- **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. 194).

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. 181).

The sections in this topic cover the following:

- Permissions Required to Use the CloudWatch Console (p. 186)
- AWS Managed (Predefined) Policies for CloudWatch (p. 188)
- Customer Managed Policy Examples (p. 189)

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:GetMetricStatisticsdata`, 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. 184) 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. 188).

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*.

Editing 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. 181) 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` Required to view alarm history.
DescribeAlarms	`cloudwatch:DescribeAlarms` Required to retrieve alarm information by name.

CloudWatch API Operations	Required Permissions (API Actions)
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.
GetMetricWidgetImage	`cloudwatch:GetMetricWidgetImage` Required to retrieve a snapshot graph of one or more CloudWatch metrics as a bitmap image.
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

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

Additionally, you can log the following AWS SDK Metrics actions in CloudTrail log files. These actions are used only by SDK Metrics and are not available for use in your code. For more information, see Monitor Applications Using AWS SDK Metrics (p. 166).

- **GetPublishingConfiguration** – Used by SDK Metrics to determine how to publish metrics.
- **GetPublishingSchema** – Used by SDK Metrics to determine how to publish metrics.
- **PutPublishingMetrics** – Used by SDK Metrics to send CloudWatch Agent health metrics to AWS Support.

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"
}
```

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.
DeleteAlarms request DescribeAlarmHistory request DescribeAlarmsForMetric request DisableAlarmActions request EnableAlarmActions request SetAlarmState request	3 transactions per second (TPS) for each of these operations. The maximum number of operation requests you can make per second without being throttled. These limits cannot be changed.
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. You can request a limit increase. 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. This limit cannot be changed.
GetMetricData	A single `GetMetricData` call can include as many as 100 `MetricDataQuery` structures. This limit cannot be changed.

Resource	Default Limit
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.

Document History

The following table describes important changes in each release of the *Amazon 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
CloudWatch agent section re-organized (p. 206)	The CloudWatch agent documentation has been rewritten to improve clarity, especially for customers using the command line to install and configure the agent. For more information, see Collecting Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent in the *Amazon CloudWatch User Guide*.	March 28, 2019
SEARCH function added to metric math expressions (p. 206)	You can now use a SEARCH function in metric math expressions. This enables you to create dashboards that update automatically as new resources are created that match the search query. For more information, see Using Search Expressions in Graphs in the *Amazon CloudWatch User Guide*.	March 21, 2019
AWS SDK Metrics for Enterprise Support (p. 206)	SDK Metrics helps you assess the health of your AWS services and diagnose latency caused by reaching your account usage limits or by a service outage. For more information, see Monitor Applications Using AWS SDK Metrics in the *Amazon CloudWatch User Guide*.	December 11, 2018
Alarms on math expressions (p. 206)	CloudWatch supports creating alarms based on metric math expressions. For more information, see Alarms on Math Expressions in the *Amazon CloudWatch User Guide*.	November 20, 2018
New CloudWatch console home page (p. 206)	Amazon has created a new home page in the CloudWatch console, which automatically displays key metrics and alarms for all the AWS services you are using. For more information, see	November 19, 2018

	Getting Started with Amazon CloudWatch in the *Amazon CloudWatch User Guide*.	
AWS CloudFormation templates for the CloudWatch Agent (p. 206)	Amazon has uploaded AWS CloudFormation templates that you can use to install and update the CloudWatch agent. For more information, see Install the CloudWatch Agent on New Instances Using AWS CloudFormation in the *Amazon CloudWatch User Guide*.	November 9, 2018
Enhancements to the CloudWatch Agent (p. 206)	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. 206)	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 Using 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. 60).	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 Collecting Metrics and Logs from	7 September 2017

Change	Description	Release Date
	Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent (p. 84).	
NAT gateway metrics	Added metrics for Amazon VPC NAT gateway.	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. 44).	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. 17).	6 July 2017
AWS Direct Connect metrics	Added metrics for AWS Direct Connect.	29 June 2017
Amazon VPC VPN metrics	Added metrics for Amazon VPC VPN.	15 May 2017
AppStream 2.0 metrics	Added metrics for AppStream 2.0.	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. 20).	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. 24).	15 February 2017
Added metrics for Amazon Polly	Added metrics for Amazon Polly.	1 December 2016
Added metrics for Amazon Kinesis Data Analytics	Added metrics for Amazon Kinesis Data Analytics.	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.	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.	20 September 2016

Change	Description	Release Date
Added metrics for Amazon API Gateway	Added metrics for Amazon API Gateway.	9 September 2016
Added metrics for AWS Key Management Service	Added metrics for AWS Key Management Service.	9 September 2016
Added metrics for the new Application Load Balancers supported by Elastic Load Balancing	Added metrics for Application Load Balancers.	11 August 2016
Added new NetworkPacketsIn and NetworkPacketsOut metrics for Amazon EC2	Added new NetworkPacketsIn and NetworkPacketsOut metrics for Amazon EC2.	23 March 2016
Added new metrics for Amazon EC2 Spot fleet	Added new metrics for Amazon EC2 Spot fleet.	21 March 2016
Added new CloudWatch Logs metrics	Added new CloudWatch Logs metrics.	10 March 2016
Added Amazon Elasticsearch Service and AWS WAF metrics and dimensions	Added Amazon Elasticsearch Service and AWS WAF metrics and dimensions.	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. 17).	8 October 2015
Added AWS Lambda metrics and dimensions	Added AWS Lambda metrics and dimensions.	4 September 2015
Added Amazon Elastic Container Service metrics and dimensions	Added Amazon Elastic Container Service metrics and dimensions.	17 August 2015

Change	Description	Release Date
Added Amazon Simple Storage Service metrics and dimensions	Added Amazon Simple Storage Service metrics and dimensions.	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. 75).	23 July 2015
Added Amazon WorkSpaces metrics and dimensions	Added Amazon WorkSpaces metrics and dimensions.	30 April 2015
Added Amazon Machine Learning metrics and dimensions	Added Amazon Machine Learning metrics and dimensions.	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. 75).	12 March 2015
Added Amazon CloudFront and Amazon CloudSearch metrics and dimensions	Added Amazon CloudFront and Amazon CloudSearch metrics and dimensions.	6 March 2015
Added Amazon Simple Workflow Service metrics and dimensions	Added Amazon Simple Workflow Service metrics and dimensions.	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. 200).	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.	16 July 2013
Added Amazon Route 53 metrics and dimensions	Added Amazon Route 53 metrics and dimensions.	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. 75).	8 January 2013
Updated EBS metrics	Updated the EBS metrics to include two new metrics for Provisioned IOPS volumes.	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. 81).	10 May 2012
New metrics	You can now access six new Elastic Load Balancing metrics that provide counts of various HTTP response codes.	19 October 2011
New feature	You can now access metrics from Amazon EMR.	30 June 2011
New feature	You can now access metrics from Amazon Simple Notification Service and Amazon Simple Queue Service.	14 July 2011
New Feature	Added information about using the `PutMetricData` API to publish custom metrics. For more information, see Publishing Custom Metrics (p. 44).	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
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