
Amazon CloudWatch

Developer Guide

API Version 2010-08-01



Amazon CloudWatch: Developer Guide

Copyright © 2014 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, Cloudfront, CloudTrail, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon CloudWatch?	1
Amazon CloudWatch Architecture	1
Amazon CloudWatch Concepts	2
Metrics	3
Namespaces	3
Dimensions	4
Time Stamps	5
Units	6
Statistics	6
Periods	7
Aggregation	7
Alarms	8
Regions	8
Supported AWS Services	9
Accessing CloudWatch	10
Regions and Endpoints	11
CloudWatch Limits	11
Related AWS Services	12
Resources	12
Getting Set Up	13
Sign Up for Amazon Web Services (AWS)	13
Sign in to the Amazon CloudWatch Console	13
Set Up the Command Line Interface	15
Getting Started with Amazon CloudWatch	16
Scenario: Monitor Your Estimated Charges Using CloudWatch	16
Step 1: Enable Monitoring of Your Estimated Charges	17
Step 2: Create a Billing Alarm	17
Step 3: Check Alarm Status	21
Step 4: Edit a Billing Alarm	21
Step 5: Delete a Billing Alarm	22
Scenario: Publish Metrics to CloudWatch	23
Step 1: Define the Data Configuration	23
Step 2: Add Metrics to CloudWatch	24
Step 3: Get Statistics From CloudWatch	24
Step 4: View Graphs with the Console	25
Viewing, Graphing, and Publishing Metrics	26
View Available Metrics	26
AWS Management Console	27
Command Line Tools	28
Query API	29
Search for Available Metrics	30
Select and Deselect Metrics	31
Get Statistics for a Metric	34
Get Statistics for a Specific EC2 Instance	34
Aggregating Statistics Across Instances	39
Get Statistics Aggregated by Auto Scaling Group	44
Get Statistics Aggregated by Image (AMI) ID	47
Graph Metrics	51
Graph a Metric	52
Graph a Metric Across Resources	53
Graph Several Metrics	55
Modify the Date and Time on a Graph	56
Modify the Statistic for a Graph	57
Modify the Period for a Graph	58
Modify a Graph's Title	59

Create an Alarm from a Metric on a Graph	61
Zoom in to a Graph	62
Move Backwards in Time on a Graph	63
Move Forwards in Time on a Graph	64
Jump to "Now" on a Graph	65
Switch the Y-Axis for a Metric	66
Save a Graph	67
Publish Custom Metrics	68
Publish Single Data Points	69
Publish Statistic Sets	70
Publish the Value Zero	70
Creating Alarms	71
Set Up Amazon Simple Notification Service	73
AWS Management Console	73
Command Line Tools	77
Create an Alarm	78
Send Email Based on CPU Usage Alarm	80
AWS Management Console	80
Command Line Tools	82
Send Email Based on Load Balancer Alarm	83
AWS Management Console	83
Command Line Tools	85
Send Email Based on Storage Throughput Alarm	85
AWS Management Console	85
Command Line Tools	87
Create Alarms That Stop or Terminate an Instance	88
Adding Actions to Amazon CloudWatch Alarms	88
Amazon CloudWatch Alarm Action Scenarios	99
Monitor Your Estimated Charges	103
Enabling the Monitoring of Your Estimated Charges	104
Creating a Billing Alarm	104
Editing a Billing Alarm	110
Checking Alarm Status	110
Deleting a Billing Alarm	111
Monitoring Log Files	113
Concepts	113
Getting Started	114
CloudWatch Logs Agent Prerequisites	114
Quick Start: Install and Configure the CloudWatch Logs Agent on an Existing EC2 Instance	115
Quick Start: Install and Configure the CloudWatch Logs Agent on a New EC2 Instance	118
Quick Start: Install the CloudWatch Logs Agent Using AWS OpsWorks and Chef	121
Quick Start: Install the CloudWatch Logs Agent Using AWS CloudFormation	125
Report the CloudWatch Logs Agent's Status	126
Start the CloudWatch Logs Agent	126
Stop the CloudWatch Logs Agent	126
CloudWatch Logs Agent Reference	127
Viewing Log Data	131
Changing Log Retention	131
Monitoring Log Data	132
Filter and Pattern Syntax	132
Creating Metric Filters	136
Listing Metric Filters	144
Deleting a Metric Filter	145
Logging API Calls	147
CloudWatch Information in CloudTrail	147
Understanding Amazon CloudWatch Log File Entries	148
Monitoring Scripts for Amazon EC2 Instances	153

Amazon CloudWatch Monitoring Scripts for Linux	153
Prerequisites	154
Getting Started	155
Using the Scripts	156
Viewing Your Custom Metrics in the AWS Management Console	160
Amazon CloudWatch Monitoring Scripts for Windows	160
Getting Started	161
Using the Scripts	162
Controlling User Access to Your AWS Account	170
Amazon CloudWatch ARNs	170
CloudWatch Actions	171
CloudWatch Keys	171
Example Policies for CloudWatch	172
Namespaces, Dimensions, and Metrics Reference	175
AWS Namespaces	176
Auto Scaling Dimensions and Metrics	176
Auto Scaling Instance Support	177
Auto Scaling Group Support	178
AWS Billing Dimensions and Metrics	179
AWS Billing Metrics	179
Dimensions for AWS Billing Metrics	180
Amazon CloudFront Dimensions and Metrics	180
Amazon CloudFront Metrics	180
Dimensions for CloudFront Metrics	181
DynamoDB Dimensions and Metrics	181
DynamoDB Metrics	181
Dimensions for DynamoDB Metrics	185
Amazon ElastiCache Dimensions and Metrics	186
Dimensions for ElastiCache Metrics	186
Host-Level Metrics	186
Metrics for Memcached	187
Metrics for Redis	189
Amazon EBS Dimensions and Metrics	191
Amazon EBS Metrics	191
Dimensions for Amazon EBS Metrics	192
Amazon Elastic Compute Cloud Dimensions and Metrics	192
Amazon EC2 Metrics	192
Dimensions for Amazon EC2 Metrics	195
Elastic Load Balancing Dimensions and Metrics	196
Elastic Load Balancing Metrics	196
Dimensions for Elastic Load Balancing Metrics	197
Amazon Elastic MapReduce Dimensions and Metrics	198
Amazon EMR Metrics	198
Amazon EMR Dimensions	202
Amazon Kinesis Dimensions and Metrics	202
Amazon Kinesis Metrics	202
Dimensions for Amazon Kinesis Metrics	203
AWS OpsWorks Dimensions and Metrics	203
AWS OpsWorks Metrics	203
Dimensions for AWS OpsWorks Metrics	204
Amazon Redshift Dimensions and Metrics	205
Amazon Redshift Metrics	205
Dimensions for Amazon Redshift Metrics	207
Amazon RDS Dimensions and Metrics	208
Amazon RDS Metrics	208
Dimensions for RDS Metrics	209
Amazon Route 53 Dimensions and Metrics	210
Amazon Route 53 Metrics	210

Dimensions for Amazon Route 53 Metrics	210
Amazon Simple Notification Service Dimensions and Metrics	210
Amazon Simple Notification Service Metrics	210
Dimensions for Amazon Simple Notification Service Metrics	211
Amazon SQS Dimensions and Metrics	211
Amazon SQS Metrics	212
Dimensions for Amazon SQS Metrics	213
Amazon SWF Dimensions and Metrics	213
Workflow Metrics	213
Activity Metrics	214
AWS Storage Gateway Dimensions and Metrics	214
AWS Storage Gateway Metrics	214
Dimensions for AWS Storage Gateway Metrics	218
Making API Requests	219
Amazon CloudWatch Endpoints	219
Query Parameters	219
The RequestId	220
Query API Authentication	220
Query API Examples Using Signature Version 2	221
Query API Error Messages Using Signature Version 2	224
Available Libraries	225
Document History	226
AWS Glossary	230

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

The rest of this section introduces the key concepts and terms that will help you understand what you need to do to monitor your resources and applications with CloudWatch.

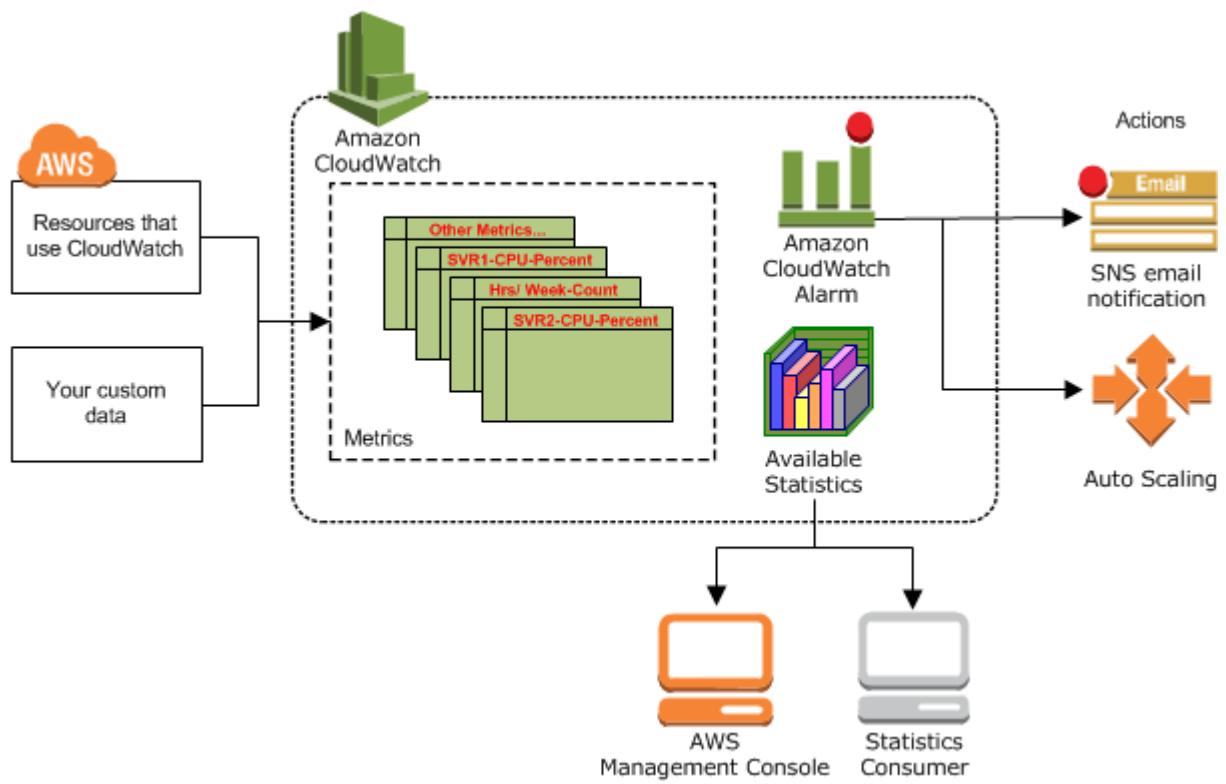
Topics

- [Amazon CloudWatch Architecture \(p. 1\)](#)
- [Amazon CloudWatch Concepts \(p. 2\)](#)
- [Supported AWS Services \(p. 9\)](#)
- [Accessing CloudWatch \(p. 10\)](#)
- [Regions and Endpoints \(p. 11\)](#)
- [CloudWatch Limits \(p. 11\)](#)
- [Related AWS Services \(p. 12\)](#)
- [Amazon CloudWatch Resources \(p. 12\)](#)

The *Getting Set Up with CloudWatch* section walks you through the process of signing up for AWS and setting up the CloudWatch command-line interface (CLI). The *Getting Started with CloudWatch* section walks you through the process of publishing metrics, getting statistics, and setting alarms.

Amazon CloudWatch Architecture

Amazon CloudWatch is basically a metrics repository. An AWS product—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 those statistics as well.



You can use metrics to calculate statistics and present the data graphically in the CloudWatch console. For more information about the other AWS resources that generate and send metrics to CloudWatch, see [Amazon CloudWatch Namespaces, Dimensions, and Metrics Reference \(p. 175\)](#).

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 Auto Scaling and Amazon Simple Notification Service (Amazon SNS) actions on your behalf. For more information about creating CloudWatch alarms, see [Alarms \(p. 8\)](#).

Amazon CloudWatch Concepts

The terminology and concepts that are central to your understanding and use of Amazon CloudWatch are described below.

Topics

- [Metrics \(p. 3\)](#)
- [Namespaces \(p. 3\)](#)
- [Dimensions \(p. 4\)](#)
- [Time Stamps \(p. 5\)](#)
- [Units \(p. 6\)](#)
- [Statistics \(p. 6\)](#)
- [Periods \(p. 7\)](#)
- [Aggregation \(p. 7\)](#)
- [Alarms \(p. 8\)](#)
- [Regions \(p. 8\)](#)

Metrics

A metric is the fundamental concept in CloudWatch and represents a time-ordered set of data points. These data points can be either your custom metrics or metrics from other services in AWS. You or AWS products publish metric data points into CloudWatch and you retrieve statistics about those data points as an ordered set of time-series data.

Think of a metric as a variable to monitor, and the data points represent the values of that variable over time. For example, the CPU usage of a particular Amazon EC2 instance is one metric, and the latency of an Elastic Load Balancing load balancer is another.

The data points themselves can come from any application or business activity from which you collect data, not just Amazon Web Services products and applications. For example, a metric might be the CPU usage of a particular Amazon EC2 instance or the temperature in a refrigeration facility.

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

You can use the `PutMetricData` API action (or the `put-metric-data` command) to create a custom metric and publish data points for it. You can add the data points in any order, and at any rate you choose. CloudWatch aggregates data points that are fully identical (duplicate values, time stamps, and units) when you request statistics on them.

CloudWatch stores your metric data for two weeks. You can publish metric data from multiple sources, such as incoming network traffic from dozens of different Amazon EC2 instances, or requested page views from several different web applications. You can request statistics on metric data points that occur within a specified time window.

Related Topics

- [PutMetricData \(put-metric-data\)](#)
- [ListMetrics \(list-metrics\)](#)
- [GetMetricStatistics \(get-metric-statistics\)](#)
- [View Available Metrics \(p. 26\)](#)

Namespaces

CloudWatch namespaces are containers for metrics. Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.

Namespace names are strings you define when you create a metric. The names must be valid XML characters, typically containing the alphanumeric characters "0-9A-Za-z" plus ". " (period), " - " (hyphen), "_ " (underscore), " / " (slash), "# " (hash), and " : " (colon). AWS namespaces all follow the convention `AWS/<service>`, such as `AWS/EC2` and `AWS/ELB`.

Note

Namespace names must be fewer than 256 characters in length.

There is no default namespace. You must specify a namespace for each data element you put into CloudWatch.

Related Topics

- [AWS Namespaces \(p. 176\)](#)

- Aggregating Statistics Across Instances (p. 39)

Dimensions

A dimension is a name/value pair that helps you to uniquely identify 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 metric.

You specify dimensions when you create a metric with the `PutMetricData` action (or its command line equivalent `put-metric-data`). Services in AWS that feed data to CloudWatch also attach dimensions to each metric. You can use dimensions to filter result sets that CloudWatch queries return.

For example, you can get statistics for a specific Amazon EC2 instance by calling `GetMetricStatistics` with the `InstanceId` dimension set to a specific Amazon EC2 instance ID.

For metrics produced by certain services such as Amazon EC2, CloudWatch can aggregate data across dimensions. For example, if you call `GetMetricStatistics` for a metric in the AWS/EC2 namespace and do not specify any dimensions, CloudWatch aggregates all data for the specified metric to create the statistic that you requested. However, CloudWatch does not aggregate across dimensions for metrics that you create with `PutMetricData` or `put-metric-data`.

Note

You can assign up to ten dimensions to a metric.

In the figure at the end of this section, the four calls to `put-metric-data` create four distinct metrics. If you make only those four calls, you could retrieve statistics for these four dimension combinations:

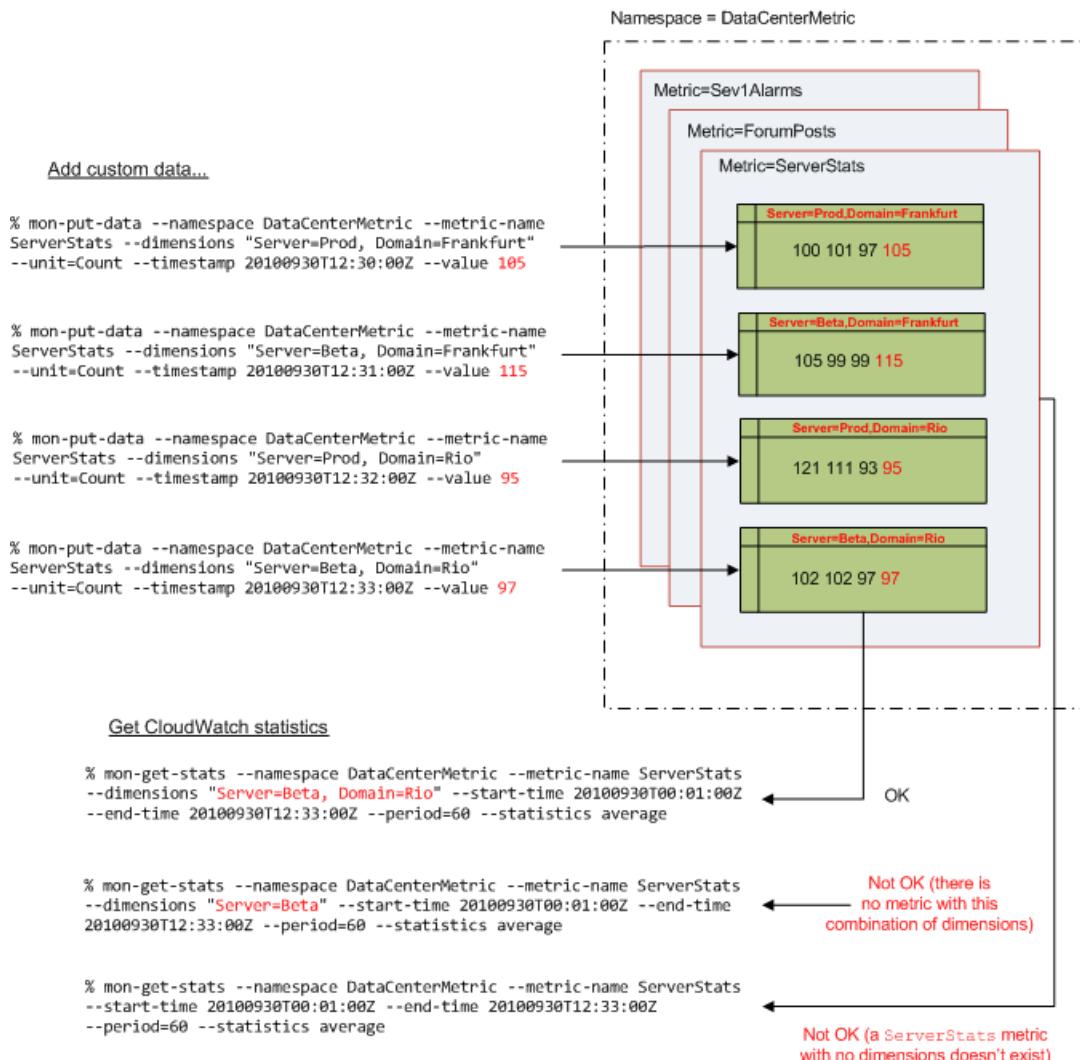
- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`
- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

You could not retrieve statistics using combinations of dimensions that you did not specifically create. For example, you could not retrieve statistics for any of the following combinations of dimensions unless you create new metrics that specify these combinations with additional calls to `put-metric-data`:

- `Server=Prod,Domain=<null>`
- `Server=<null>,Domain=Frankfurt`
- `Server=Beta,Domain=<null>`
- `Server=<null>,Domain=Rio`
- `Server=Prod`
- `Server=Beta`

Important

CloudWatch treats each unique combination of dimensions as a separate metric. For example, each call to `put-metric-data` in the following figure creates a separate metric because each call uses a different set of dimensions. This is true even though all four calls use the same metric name (`ServerStats`). For information on how this affects pricing, see the [Amazon CloudWatch product information page](#).



Related Topics

- [put-metric-data](#)
- [get-metric-statistics](#)
- [Dimensions for Amazon EC2 Metrics \(p. 195\)](#)
- [Dimensions for Elastic Load Balancing Metrics \(p. 197\)](#)
- [Dimensions for RDS Metrics \(p. 209\)](#)

Time Stamps

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

The time stamp you use in the request must be a `dateTime` object, with the complete date plus hours, minutes, and seconds. For more information, see <http://www.w3.org/TR/xmlschema-2/#dateTime>. For example: `2013-01-31T23:59:59Z`. Although it is not required, we recommend you provide the time stamp

in the Coordinated Universal Time (UTC or Greenwich Mean Time) time zone. When you retrieve your statistics from CloudWatch, all times reflect the UTC time zone.

Units

Units represent your statistic's unit of measure. For example, the units for the Amazon EC2 NetworkIn metric are *Bytes* because NetworkIn tracks the number of bytes that an instance receives on all network interfaces.

You can also specify a unit when you create a custom metric. Units help provide conceptual meaning to your data. Metric data points that specify a unit of measure, such as *Percent*, are aggregated separately. The following list provides some of the more common units that CloudWatch supports:

- Seconds
- Bytes
- Bits
- Percent
- Count
- Bytes/Second (bytes per second)
- Bits/Second (bits per second)
- Count/Second (counts per second)
- None (default when no unit is specified)

For a complete list of the units that CloudWatch supports, see the [MetricDatum](#) data type in the [Amazon CloudWatch API Reference](#).

Though CloudWatch attaches no significance to a unit internally, other applications can derive semantic information based on the unit you choose. When you publish data without specifying a unit, CloudWatch associates it with the *None* unit. 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 will be returned, one for each unit.

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 services in AWS 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 <code>Sum / SampleCount</code> during the specified period. By comparing this statistic with the <code>Minimum</code> and <code>Maximum</code> , you can determine the full scope of a metric and how close the average use is to the <code>Minimum</code> and <code>Maximum</code> . This comparison helps you to know when to increase or decrease your resources as needed.

Statistic	Description
SampleCount	The count (number) of data points used for the statistical calculation.

You use the `GetMetricStatistics` API action or the `get-metric-statistics` command to retrieve statistics, specifying the same values that you used for the namespace, metric name, and dimension parameters when the metric values were created. You also specify the start and end times that CloudWatch will use for the aggregation. The starting and ending points can be as close together as 60 seconds, and as far apart as two weeks.

Amazon CloudWatch allows you to add pre-calculated statistics using the `PutMetricData` API action (or the `put-metric-data` command) with the `StatisticValues (statistic-values)` parameter. 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.

Related Topics

- [PutMetricData \(put-metric-data\)](#)
- [GetMetricStatistics \(get-metric-statistics\)](#)

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. Although periods are expressed in seconds, the minimum granularity for a period is one minute. Accordingly, you specify period values as multiples of 60. For example, to specify a period of six minutes, you would use the value 360. You can adjust how the data is aggregated by varying the length of the period. A period can be as short as one minute (60 seconds) or as long as two weeks (1,209,600 seconds).

When you call `GetMetricStatistics`, you can specify the period length with the `Period` parameter. Two related parameters, `StartTime` and `EndTime`, determine the overall length of time associated with the statistics. The default value for the `Period` parameter is 60 seconds, whereas the default values for `StartTime` and `EndTime` give you the last hour's worth of statistics.

The values you select for the `StartTime` and `EndTime` parameters determine how many periods `GetMetricStatistics` will return. For example, calling `GetMetricStatistics` with the default values for the `Period`, `EndTime`, and `StartTime` parameters returns an aggregated set of statistics for each minute of the previous hour. If you prefer statistics aggregated into ten-minute blocks, set `Period` to 600. For statistics aggregated over the entire hour, use a `Period` value of 3600.

Periods are also an important part of the CloudWatch alarms feature. When you create an alarm to monitor a specific metric, you are asking CloudWatch to compare that metric to the threshold value that you supplied. 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 consecutive periods the threshold must be breached before you are notified. For more information about alarms, see [Alarms \(p. 8\)](#).

Aggregation

Amazon CloudWatch aggregates statistics according to the period length that you specify in calls to `GetMetricStatistics`. You can publish as many data points as you want with the same or similar time stamps. CloudWatch aggregates them by period length when you get statistics about those data points with `GetMetricStatistics`. 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. Subsequent calls to `GetMetricStatistics` returns aggregated statistics about those data points. You can even do this in one `PutMetricData` request. CloudWatch accepts multiple data points in the same `PutMetricData` call with the same time stamp. You can also publish multiple data points for the same or different metrics, with any time stamp. The size of a `PutMetricData` request, however, is limited to 8KB for HTTP GET requests and 40KB for HTTP POST requests. You can include a maximum of 20 data points in one `PutMetricData` request.

For large data sets that would make the use of `PutMetricData` impractical, CloudWatch allows for the insertion of a pre-aggregated data set called a *StatisticSet*. With StatisticSets you give CloudWatch the Min, Max, Sum, and SampleCount of a number of data points. StatisticSets is commonly used when you need to collect data many times in a minute. For example, let's say you have a metric for the request latency of a web page. It doesn't make sense to do a `PutMetricData` request with every web page hit. We suggest you collect the latency of all hits to that web page, aggregate them together once a minute and send that StatisticSet 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.

Alarms

Alarms can automatically initiate actions on your behalf, based on parameters you specify. An alarm watches a single metric over a specified time period, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Auto Scaling policy. Alarms invoke actions for sustained state changes only. CloudWatch alarms will 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 instances provides metrics 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 instances provides metrics every 1 minute; when setting an alarm on a Detailed monitoring metric, select a period of at least 60 seconds (1 minute).

Related Topics

- [PutMetricAlarm](#)
- [put-metric-alarm](#)
- [Creating Amazon CloudWatch Alarms \(p. 71\)](#)

For examples that show you how to set up CloudWatch alarms that invoke an Auto Scaling policy and an Amazon SNS topic, see [Creating Amazon CloudWatch Alarms \(p. 71\)](#).

Regions

Amazon 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*. Regions are large and widely dispersed geographic locations.

Each Amazon Region is designed to be completely isolated from the other Amazon Regions. This achieves the greatest possible failure isolation and stability, and it makes the locality of each Amazon resource

unambiguous. Amazon CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions.

For more information about the endpoints that represent each region, see [Regions and Endpoints](#) in the [Amazon Web Services General Reference](#).

Supported AWS Services

CloudWatch monitors the following services. As soon as you begin using a service, it automatically sends metrics to CloudWatch for you.

CloudWatch offers either basic or detailed monitoring for supported AWS products. Basic monitoring means that a service sends data points to CloudWatch every five minutes. Detailed monitoring means that a service sends data points to CloudWatch every minute.

Note

If you are using a service that supports both basic and detailed data collection (for example, Amazon EC2 and Auto Scaling), and you want to access detailed statistics, you must enable detailed metric collection for that service.

- **Auto Scaling**

Auto Scaling sends data to CloudWatch every 5 minutes by default. For an additional charge, you can enable detailed monitoring for Auto Scaling, which sends data to CloudWatch every minute. You can create alarms using [Auto Scaling Dimensions and Metrics \(p. 176\)](#). For more information, see [Monitor Your Auto Scaling Instances](#) in the [Auto Scaling Developer Guide](#).

- **Amazon DynamoDB**

Amazon DynamoDB sends data to CloudWatch every 5 minutes. You can create alarms using [DynamoDB Dimensions and Metrics \(p. 181\)](#). For more information, see [Monitoring DynamoDB Tables with Amazon CloudWatch](#) in the [Amazon DynamoDB Developer Guide](#).

- **Amazon ElastiCache**

Amazon ElastiCache sends data to CloudWatch every minute. You can create alarms using [Amazon ElastiCache Dimensions and Metrics \(p. 186\)](#). For more information, see [Viewing Cache Cluster and Cache Node Metrics](#) in the [Amazon ElastiCache User Guide](#).

- **Amazon Elastic Block Store**

Amazon Elastic Block Store sends data to CloudWatch every 5 minutes. You can create alarms using [Amazon EBS Dimensions and Metrics \(p. 191\)](#). For more information, see [Monitoring the Status of Your Volumes](#) in the [Amazon EC2 User Guide for Linux Instances](#).

- **Amazon Elastic Compute Cloud**

Amazon EC2 sends data to CloudWatch every 5 minutes by default. For an additional charge, you can enable detailed monitoring for Amazon EC2, which sends data to CloudWatch every minute. You can create alarms using [Amazon Elastic Compute Cloud Dimensions and Metrics \(p. 192\)](#). For more information, see [Monitoring Your Instances](#) in the [Amazon EC2 User Guide for Linux Instances](#).

- **Elastic Load Balancing**

Elastic Load Balancing sends data to CloudWatch every minute. You can create alarms using [Elastic Load Balancing Dimensions and Metrics \(p. 196\)](#). For more information, see [Monitor Your Load Balancer Using Amazon CloudWatch](#) in the [Elastic Load Balancing Developer Guide](#).

- **Amazon Elastic MapReduce**

Amazon Elastic MapReduce sends data to CloudWatch every 5 minutes. You can create alarms using [Amazon Elastic MapReduce Dimensions and Metrics \(p. 198\)](#). For more information, see [Monitor Metrics with Amazon CloudWatch](#) in the *Amazon Elastic MapReduce Developer Guide*.

- **Amazon Kinesis**

Amazon Kinesis sends data to CloudWatch every minute. You can create alarms using [Amazon Kinesis Dimensions and Metrics \(p. 202\)](#). For more information, see [Monitoring Amazon Kinesis with Amazon CloudWatch](#) in the *Amazon Kinesis Developer Guide*.

- **AWS OpsWorks**

AWS OpsWorks sends data to CloudWatch every minute. You can create alarms using [AWS OpsWorks Dimensions and Metrics \(p. 203\)](#). For more information, see [Monitoring](#) in the *AWS OpsWorks User Guide*.

- **Amazon Redshift**

Amazon Redshift sends data to CloudWatch every minute. You can create alarms using [Amazon Redshift Dimensions and Metrics \(p. 205\)](#). For more information, see [Monitoring Amazon Redshift Cluster Performance](#) in the *Amazon Redshift Cluster Management Guide*.

- **Amazon Relational Database Service**

Amazon Relational Database Service sends data to CloudWatch every minute. You can create alarms using [Amazon RDS Dimensions and Metrics \(p. 208\)](#). For more information, see [Monitoring a DB Instance](#) in the *Amazon Relational Database Service User Guide*.

- **Amazon Route 53**

Amazon Route 53 sends data to CloudWatch every minute. You can create alarms using [Amazon Route 53 Dimensions and Metrics \(p. 210\)](#). For more information, see [Monitoring Health Checks Using Amazon CloudWatch](#) in the *Amazon Route 53 Developer Guide*.

- **Amazon Simple Notification Service**

Amazon Simple Notification Service sends data to CloudWatch every 5 minutes. You can create alarms using [Amazon Simple Notification Service Dimensions and Metrics \(p. 210\)](#). For more information, see [Monitoring Amazon SNS with Amazon CloudWatch](#) in the *Amazon Simple Notification Service Developer Guide*.

- **Amazon Simple Queue Service**

Amazon Simple Queue Service sends data to CloudWatch every 5 minutes. You can create alarms using [Amazon SQS Dimensions and Metrics \(p. 211\)](#). For more information, see [Monitoring Amazon SQS with Amazon CloudWatch](#) in the *Amazon Simple Queue Service Developer Guide*.

- **Amazon Simple Workflow Service**

Amazon Simple Workflow Service sends data to CloudWatch every 5 minutes. You can create alarms using [Amazon SWF Dimensions and Metrics \(p. 213\)](#). For more information, see [Viewing Amazon SWF Metrics for CloudWatch using the AWS Management Console](#); in the *Amazon Simple Workflow Service Developer Guide*.

- **AWS Storage Gateway**

AWS Storage Gateway sends data to CloudWatch every 5 minutes. You can create alarms using [AWS Storage Gateway Dimensions and Metrics \(p. 214\)](#). For more information, see [Monitoring Your AWS Storage Gateway](#) in the *AWS Storage Gateway User Guide*.

Accessing CloudWatch

You can access CloudWatch using any of the following:

- Amazon CloudWatch console

For more information about the CloudWatch console, see [Sign in to the Amazon CloudWatch Console \(p. 13\)](#).

- AWS Console for Android and iOS

For more information about the AWS Console, see [AWS Console for Android and iOS](#).

- CloudWatch CLI

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

- 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*.

- CloudWatch API

For more information about the CloudWatch API, see [Amazon CloudWatch API Reference](#).

- AWS SDKs

For more information about the AWS SDKs, see [Tools for Amazon Web Services](#).

Regions and Endpoints

You monitor metrics and create alarms in a specific AWS region. You send your CloudWatch requests to a region-specific endpoint. For a list of supported AWS regions, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

CloudWatch Limits

CloudWatch has the following limits:

- You get 10 CloudWatch metrics, 10 alarms, 1,000,000 API requests, and 1,000 Amazon SNS email notifications per customer per month for free.
- You can assign up to 10 dimensions per metric.
- You can create up to 5000 alarms per AWS account.
- Metric data is kept for 2 weeks.
- The size of a [PutMetricData](#) request is limited to 8KB for HTTP GET requests and 40KB for HTTP POST requests.
- You can include a maximum of 20 [MetricDatum](#) items in one [PutMetricData](#) request. A MetricDatum can contain a single value or a [StatisticSet](#) representing many values.

CloudWatch Logs has the following limits:

- The maximum number of log groups per AWS account is 500.
- The maximum number of metric filters is 100 per log group.
- The maximum rate of a [PutLogEvents](#) request is 5 requests per second per log stream. Since the maximum batch size of a PutLogEvents request is 32KB, this means that uploads to a single log stream are limited to a maximum rate of 160KB/s.
- The maximum rate of a [GetLogEvents](#) request is 10 requests per second per AWS account.

Related AWS Services

The following services are used in conjunction with CloudWatch:

- Auto Scaling is a web service that enables you to automatically launch or terminate Amazon Elastic Compute Cloud (Amazon EC2) instances based on user-defined policies, health status checks, and schedules. You can use a CloudWatch alarm with Auto Scaling to scale Amazon EC2 instances based on demand. For more information, see [Scale Based on Demand](#) in the *Auto Scaling Developer Guide*.
- Amazon Simple Notification Service (Amazon SNS) is a web service that 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 Simple Notification Service \(p. 73\)](#).
- AWS CloudTrail is a web service that enables you to monitor the calls made to the Amazon CloudWatch API for your account, including calls made by the AWS Management Console, command line interface (CLI), and other services. When CloudTrail logging is turned on, CloudWatch will write log files into the Amazon S3 bucket that you specified when you configured CloudTrail. Each log file can contain one or more records, depending on how many actions must be performed to satisfy a request. For more information about AWS CloudTrail, see [What is AWS CloudTrail?](#) in the *AWS CloudTrail User Guide*. For an example of the type of data that CloudWatch writes into CloudTrail log files, see [Logging Amazon CloudWatch API Calls in AWS CloudTrail \(p. 147\)](#).

Amazon CloudWatch Resources

The following table lists related resources that you'll find useful as you work with Amazon CloudWatch.

Resource	Description
Amazon CloudWatch Technical FAQ	The FAQ covers the top 20 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 samples, 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 products 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 web page 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'll need an AWS account to use services (e.g., Amazon EC2) to generate metrics that you can view in the CloudWatch console, a point-and-click web-based interface. In addition, you'll need to install and configure the AWS command line interface (CLI).

Topics

- [Sign Up for Amazon Web Services \(AWS\) \(p. 13\)](#)
- [Sign in to the Amazon CloudWatch Console \(p. 13\)](#)
- [Set Up the Command Line Interface \(p. 15\)](#)

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 <http://aws.amazon.com>, and then click **Sign Up**.
2. Follow the on-screen instructions.

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

Sign in to the Amazon CloudWatch Console

To sign in to the Amazon CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.

The monitoring dashboard opens. Your dashboard might look something like the following:

The screenshot shows the CloudWatch Dashboard. On the left, a sidebar lists 'Dashboard', 'Alarms' (with three items: ALARM, INSUFFICIENT, OK), 'Billing', and 'Metrics'. The main area has three sections: 'Metric Summary' (empty), 'Alarm Summary' (empty), and 'Service Health' (showing one service is operating normally). A navigation bar at the bottom includes links for 'Create Alarm', 'Feedback', and legal notices.

If you do not have any alarms, the **Your Alarms** section will have a **Create Alarm** button. Even if this is the first time you are using the CloudWatch console, the **Your Metrics** section could already report that you are using a significant number of metrics, because several AWS products push free metrics to Amazon CloudWatch automatically.

2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



Set Up the Command Line Interface

You can use the Amazon CloudWatch command line interface (CLI) or the AWS CLI with CloudWatch. Before you can use either CLI, however, you have to install and configure them.

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

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*.

Getting Started with Amazon CloudWatch

You can learn how to use Amazon CloudWatch by completing the following scenarios. In the first scenario, you'll use the CloudWatch console, a point-and-click web-based interface, to create a billing alarm that tracks your AWS usage and lets you know when you've exceeded a certain spending threshold. In the second, more advanced scenario, you'll use the AWS command line interface (CLI) to publish a single metric for a hypothetical application named *GetStarted*.

- Scenario: Monitor Your Estimated Charges Using CloudWatch (p. 16)
- Scenario: Publish Metrics to CloudWatch (p. 23)

Scenario: Monitor Your Estimated Charges Using CloudWatch

In this scenario, you'll create an CloudWatch alarm that will monitor your estimated Amazon Web Services (AWS) 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 that is stored for 14 days. Billing metric data is stored in the US East (N. Virginia) region and represent 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 are sent using Amazon Simple Notification Service (Amazon SNS) notification.

Topics

- Step 1: Enable Monitoring of Your Estimated Charges (p. 17)
- Step 2: Create a Billing Alarm (p. 17)
- Step 3: Check Alarm Status (p. 21)
- Step 4: Edit a Billing Alarm (p. 21)
- Step 5: Delete a Billing Alarm (p. 22)

Step 1: Enable Monitoring of Your Estimated Charges

Before you can create an alarm on your estimated charges, you must enable monitoring of your estimated AWS charges, which creates metric data that you can use to create a billing alarm. It takes about 15 minutes before you can view billing data and create alarms. After you enable billing metrics you cannot disable the collection of data, but you can delete any alarms you have created. You must be signed in as the account owner (the "root user") to enable billing alerts for your AWS account.

1. Open the Billing and Cost Management console at <https://console.aws.amazon.com/billing/home?#>.
2. In the spaces provided, enter your user name and password, and then click **Sign in using our secure server**.
3. In the navigation pane, click **Preferences**, and then select the **Receive Billing Alerts** check box.

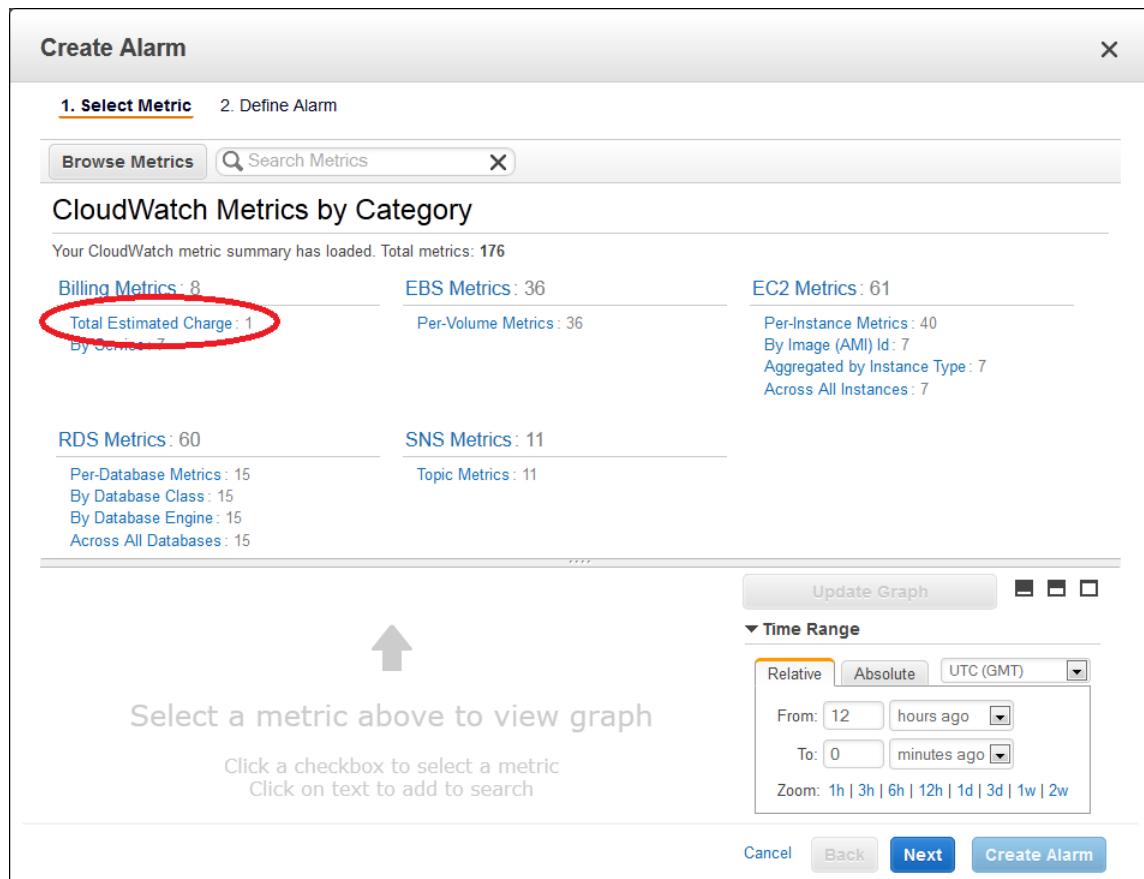
The screenshot shows the 'Preferences' section of the AWS Billing and Cost Management console. On the left, a navigation pane lists options like Dashboard, Bills, Payment Methods, Payment History, Consolidated Billing, Account Settings, Reports, **Preferences** (which is selected and highlighted with an orange bar), and Credits. The main area is titled 'Preferences' and contains three sections: 'Receive PDF Invoice By Email', 'Receive Billing Alerts', and 'Receive Billing Reports'. The 'Receive Billing Alerts' section is highlighted with a red border. It includes a description: 'Turn on this feature to monitor your AWS usage charges and recurring fees automatically, making it easier to track and manage your spending on AWS. You can set up billing alerts to receive email notifications when your charges reach a specified threshold. Once enabled, this preference cannot be disabled.' Below this is another section for 'Receive Billing Reports' with a note: 'Turn on this feature to receive ongoing reports of your AWS charges once or more daily. AWS delivers these reports to the Amazon S3 bucket that you specify where indicated below. For consolidated billing customers, AWS generates reports only for paying accounts. Linked accounts cannot sign up for billing reports.' At the bottom, there's a 'Save to S3 Bucket:' input field with 'bucket name' and a 'Verify' button, and a note: 'Note: You must apply appropriate permissions to your S3 bucket [sample policy](#)'.

Step 2: Create a Billing Alarm

After you've enabled monitoring of your estimated AWS charges, you can create a billing alarm in the Amazon CloudWatch console. In this scenario, you'll create an alarm that will send an email message when your estimated charges for AWS exceed \$200. When you enable the monitoring of your estimated charges for the first time, it takes about 15 minutes before you can view billing data and set billing alarms.

To create a billing alarm

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, click **Alarms**, and then in the **Alarms** pane, click **Create Alarm**.
3. In the **CloudWatch Metrics by Category** pane, under **Billing Metrics**, click **Total Estimated Charge**.



4. Under **Billing > Total Estimated Charge**, select the **EstimatedCharges** metric to view a graph of billing data in the lower pane.

Amazon CloudWatch Developer Guide

Step 2: Create a Billing Alarm

Create Alarm

1. Select Metric 2. Define Alarm

Browse Metrics X Billing > Total Estimated Charge ◀ ▶ 1 to 1 of 1 Metrics

Showing all results (1) for Billing > Total Estimated Charge.

Select All | Clear

Billing > Total Estimated Charge

Currency	Metric Name
<input checked="" type="checkbox"/> USD	EstimatedCharges

EstimatedCharges (None) Maximum 6 Hours Update Graph

Time Range

Relative Absolute UTC (GMT)

From: 12 hours ago To: 0 minutes ago

Zoom: 1h | 3h | 6h | 12h | 1d | 3d | 1w | 2w

Cancel Back Next Create Alarm

- Click **Next**, and then in **Alarm Threshold** pane, in the **Name** box, enter a unique, friendly name for the alarm (for example, My Estimated Charges).

Create Alarm

1. Select Metric 2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name: My Estimated Charges

Description: Estimated Monthly Charges

Whenever charges for: EstimatedCharges

is: \geq USD \$ 200

Actions

Define what actions are taken when your alarm changes state.

Notification Delete

Whenever this alarm: State is ALARM

Send notification to: Select a notification list New list

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line

EstimatedCharges \geq 200

Namespace: AWS/Billing **Currency:** USD **Metric Name:** EstimatedCharges

+ Notification + AutoScaling Action + EC2 Action

Cancel Back Next Create Alarm

6. In the **Description** box, enter a description for the alarm (for example, Estimated Monthly Charges).
7. Under **Whenever charges for**, in the **is** drop-down list, select **>=** (greater than or equal to), and then in the **USD** box, set the monetary amount (for example, 200) that must be exceeded to trigger the alarm and send an email.

Note

Under **Alarm Preview**, in the **Estimated Monthly Charges** thumbnail graph, you can see an estimate of your charges that you can use to set an appropriate threshold for the alarm.

8. Under **Actions**, click **Notification**, and then in the **Whenever this alarm** drop-down menu, click **State is ALARM**.
9. In the **Send notification to** box, select an existing Amazon SNS topic.

To create a new Amazon SNS topic, click **Create topic**, and then in the **Send notification to** box, enter a name for the new Amazon SNS topic (for example., CFO), and in the **Email list** box, enter the email address (for example, john.stiles@example.com) where email notifications should be sent.

Note

If you create a new Amazon SNS topic, the email account associated with the topic will receive a subscription confirmation email. You must confirm the subscription in order to receive future email notifications when the alarm is triggered.

Create Alarm

X

1. Select Metric
2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name: My Estimated Charges

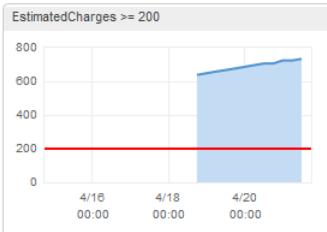
Description: Estimated Monthly Charges

Whenever charges for: EstimatedCharges

is: >= USD \$ 200

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line



EstimatedCharges >= 200

Date	EstimatedCharges
4/18 00:00	0
4/18 00:00	650
4/20 00:00	750

Actions

Define what actions are taken when your alarm changes state.

Notification

Whenever this alarm:

Send notification to: CEO

Email list: john.stiles@example.com

This notification list is managed in the SNS console.

+ Notification
+ AutoScaling Action
+ EC2 Action

Cancel
Back
Next
Create Alarm

10. Click **Create Alarm**.

Important

If you added an email address to the list of recipients or created a new topic, Amazon SNS sends a subscription confirmation email to each new address shortly after you create an

alarm. Remember to click the link contained in that message, which confirms your subscription. Alert notifications are only sent to confirmed addresses.

11. To view your billing alarm in the CloudWatch console, in the navigation pane, under **Alarms**, click **Billing**.

Step 3: Check Alarm Status

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

To check alarm status using the CloudWatch console

1. Sign in to the AWS Management Console and 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 the US East (N. Virginia) region and represent worldwide charges. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **Alarms**, click **Billing**.

Step 4: Edit a Billing Alarm

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

To edit a billing alarm using the CloudWatch console

1. Sign in to the AWS Management Console and 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 the US East (N. Virginia) region and represent worldwide charges. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **Alarms**, click **Billing**.
4. In the list of alarms, select the check box next to the alarm you want to change, and then click **Modify**.
5. Under **Alarm Threshold**, in the **USD** box, set the monetary amount (for example, 400) that must be exceeded to trigger the alarm and send an email, and then click **Save Changes**.

Step 5: Delete a Billing Alarm

Now that you've enabled billing and have created and edited your first billing alarm, you can delete the billing alarm if you no longer need it.

To delete a billing alarm using the CloudWatch console

1. Sign in to the AWS Management Console and 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 the US East (N. Virginia) region and represent worldwide charges. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **Alarms**, click **Billing**.
4. In the list of alarms, select the check box next to the alarm you want to delete, and then click **Delete**.
5. In the **Delete Alarms** dialog box, click **Yes, Delete**.

Scenario: Publish Metrics to CloudWatch

Now that you have installed the AWS CLI, you're ready to publish metrics to CloudWatch. In this scenario, you'll use the `put-metric-data` command in the AWS CLI to publish a single metric for a hypothetical application named *GetStarted*. Then, you'll use the CloudWatch console to view statistical graphs for those metrics.

For more information about the `put-metric-data` command, see [put-metric-data](#) in the *AWS Command Line Interface Reference*.

Topics

- [Step 1: Define the Data Configuration \(p. 23\)](#)
- [Step 2: Add Metrics to CloudWatch \(p. 24\)](#)
- [Step 3: Get Statistics From CloudWatch \(p. 24\)](#)
- [Step 4: View Graphs with the Console \(p. 25\)](#)

Step 1: Define the Data Configuration

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

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

- Hour one: 87, 51, 125, 235
- Hour two: 121, 113, 189, 65, 89
- Hour three: 100, 47, 133, 98, 100, 328

You can publish data to CloudWatch as single data points or as an aggregated set of data points called a *statistic set*. You'll publish the data points from hour one as single data points.

Hour	Raw Data
1	87
1	51
1	125
1	235

For the data from hours two and three, you'll aggregate the data points and publish a statistic set for each hour.

Note

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`. The key values are shown in the following table.

Hour	Raw Data	Sum	Minimum	Maximum	SampleCount
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 begin adding data.

Note

When you use the `put-metric-data` command, you must use a date range within the past two weeks. There is currently no function to delete data points. CloudWatch automatically deletes data points with a timestamp more than two weeks old.

To publish data points to CloudWatch

1. Open a command prompt and enter the following commands, but replace the time stamp with a time stamp that represents two hours in the past in Universal Coordinated Time (UTC).

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted --timestamp 2014-02-14T20:30:00Z --value 87 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted --timestamp 2014-02-14T20:30:00Z --value 51 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted --timestamp 2014-02-14T20:30:00Z --value 125 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted --timestamp 2014-02-14T20:30:00Z --value 235 --unit Milliseconds
```

The AWS CLI returns a response only when it cannot execute the command.

2. Enter the second data point, but this time use a time stamp one hour later than the first one in Universal Coordinated Time (UTC).

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted --timestamp 2014-02-14T21:30:00Z --statistic-values Sum=577,Minimum=65,Maximum=189,SampleCount=5 --unit Milliseconds
```

3. Enter the last data point, but this time omit the time stamp to get the current time (the default value).

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

After adding metrics, you can get statistics.

Step 3: Get Statistics From CloudWatch

Now that you have published metrics to CloudWatch, you are ready to retrieve statistics that are based on those metrics.

To retrieve statistics with the command line tools

- Specify a `--start-time` and `--end-time` far enough in the past to cover the earliest time stamp that you published.

```
aws cloudwatch get-metric-statistics --namespace GetStarted --metric-name RequestLatency --statistics Average --start-time 2014-02-14T00:00:00Z --end-time 2014-02-15T00:00:00Z --period 60
```

The AWS CLI returns the following json:

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

To see your statistics in a visual format, you can use the CloudWatch Console to view graphs.

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

- Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
- If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



- In the **Navigation** pane, click **Metrics**. **CloudWatch Metrics by Category** opens in the right pane.
- In the **CloudWatch Metrics by Category** pane, in the search box, type **RequestLatency**.
- Select the check box next to the **RequestLatency** metric name. A graph of the metric's data is displayed in the lower pane.
- To change the graph, choose different values from the **Statistic** and **Period** lists located next to graph's title.
- To create an alarm for this metric, under **Tools**, click **Create Alarm**.

Congratulations! You've successfully published and viewed custom metrics.

Viewing, Graphing, and Publishing Metrics

Metrics are data about the performance of your systems. By default, a set of free metrics is provided for Amazon EC2 instances, Amazon EBS volumes, Amazon RDS DB instances, and Elastic Load Balancing. You can also choose to enable detailed monitoring for your Amazon EC2 instances, or add your own application metrics. Metric data is kept for a period of two weeks enabling you to view up to the minute data and also historical data. Amazon CloudWatch can load all the metrics in your account for search, graphing and alarms with the AWS Management Console. This includes both AWS resource metrics and application metrics that you provide.

You can use the following procedures to graph metrics in CloudWatch. After you have completed these procedures, you can then create alarms for a metric. For more information, see [Creating Amazon CloudWatch Alarms \(p. 71\)](#).

Topics

- [View Available Metrics \(p. 26\)](#)
- [Search for Available Metrics \(p. 30\)](#)
- [Select and Deselect Metrics \(p. 31\)](#)
- [Get Statistics for a Metric \(p. 34\)](#)
- [Graph Metrics \(p. 51\)](#)
- [Publish Custom Metrics \(p. 68\)](#)

View Available Metrics

Only those services in AWS that you're using send metrics to Amazon CloudWatch. You can use the Amazon CloudWatch console, the `list-metrics` command, or the `ListMetrics` API to view the available metrics.

AWS Management Console

To view available metrics by category

You can view metrics by category. Metrics are grouped first by Namespace, and then by the various Dimension combinations within each Namespace. For example, you can view all EC2 metrics, or EC2 metrics grouped by instance ID, instance type, image (AMI) ID, or Auto Scaling Group.

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Metrics**.

The screenshot shows the CloudWatch Metrics by Category page. On the left, there is a navigation pane with a 'Metrics' section selected. The main area displays a grid of metric categories. Each category has a title, a count, and a detailed breakdown of metrics. For example, the 'EC2 Metrics' category has a total of 272 metrics, broken down into Per-Instance Metrics (181), By Auto Scaling Group (56), By Image (AMI) Id (14), Aggregated by Instance Type (14), and Across All Instances (7). Other categories shown include Billing Metrics (35), DynamoDB Metrics (4), ELB Metrics (95), ElastiCache Metrics (87), OpsWorks Metrics (45), RDS Metrics (104), and Redshift Metrics (26).

4. In the **CloudWatch Metrics by Category** pane, under **EC2 Metrics**, select **Per-Instance Metrics**, and then in the upper pane, scroll down to view the full list of metrics.

Browse Metrics Search Metrics EC2 > Per-Instance Metrics 1 to 50 of 181 Metrics

Showing all results (181) for EC2 > Per-Instance Metrics.

Select All | Clear

EC2 > Per-Instance Metrics

InstanceId	Metric Name
i-14d5ac6c	CPUUtilization
i-14d5ac6c	DiskReadBytes
i-14d5ac6c	DiskReadOps
i-14d5ac6c	DiskWriteBytes
i-14d5ac6c	DiskWriteOps
...	

Update Graph

Time Range

Relative Absolute UTC (GMT)

From: 12 hours ago To: 0 minutes ago

Zoom: 1h | 3h | 6h | 12h | 1d | 3d | 1w | 2w

Select a metric above to view graph

Click a checkbox to select a metric Click on text to add to search

Command Line Tools

To list available metrics across multiple Amazon EC2 instances

- Enter the `list-metrics` command.

```
Prompt>aws cloudwatch list-metrics --namespace "AWS/EC2"
```

The AWS CLI returns the following:

```
{  
  "Metrics": [  
    {  
      "Namespace": "AWS/EC2",  
      "Dimensions": [  
        {  
          "Name": "InstanceId",  
          "Value": "i-dd8855ba"  
        }  
      ],  
      "MetricName": "DiskReadBytes"  
    },  
    {  
      "Namespace": "AWS/EC2",  
      "Dimensions": [  
        {  
          "Name": "InstanceId",  
          "Value": "i-0b12b76c"  
        }  
      ],  
      "MetricName": "CPUUtilization"  
    }  
  ]}
```

```
        } ,
        {
            "Namespace": "AWS/EC2" ,
            "Dimensions": [ ] ,
            "MetricName": "NetworkIn"
        } ,
        {
            "Namespace": "AWS/EC2" ,
            "Dimensions": [
                {
                    "Name": "InstanceId" ,
                    "Value": "i-e31dbd84"
                }
            ] ,
            "MetricName": "DiskReadOps"
        } ,
        {
            "Namespace": "AWS/EC2" ,
            "Dimensions": [
                {
                    "Name": "InstanceId" ,
                    "Value": "i-13bf6574"
                }
            ] ,
            "MetricName": "DiskWriteBytes"
        } ,
        {
            "Namespace": "AWS/EC2" ,
            "Dimensions": [
                {
                    "Name": "InstanceId" ,
                    "Value": "i-2840c24f"
                }
            ] ,
            "MetricName": "DiskReadOps"
        } ,
        {
            "Namespace": "AWS/EC2" ,
            "Dimensions": [
                {
                    "Name": "InstanceId" ,
                    "Value": "i-9960c1fe"
                }
            ] ,
            "MetricName": "NetworkOut"
        }
    }
```

Query API

To determine available metrics across multiple instances

- Call [ListMetrics](#) to generate a list of all of your valid metrics.

This returns a list of metrics. An example metric might look like:

- *MetricName* = CPUUtilization
- *Dimensions* (*Name*=InstanceId, *Value*=i-5431413d)
- *Namespace* = AWS/EC2

Search for Available Metrics

You can search within all the metrics in your account using targeted search terms. Metrics are returned that have matching results within their Namespace, Metric Name, or Dimensions.

To search for available metrics in CloudWatch

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Metrics**.

The screenshot shows the 'CloudWatch Metrics by Category' pane. On the left, there is a sidebar with a 'Metrics' section selected, showing other services like Alarms, Billing, DynamoDB, EBS, EC2, ELB, ElastiCache, OpsWorks, RDS, Redshift, Route 53, SNS, and SQS. Each service has a count of metrics: ALARM (10), INSUFFICIENT (48), OK (29), Billing (1), DynamoDB (35), EBS (80), EC2 (272), ELB (95), ElastiCache (87), OpsWorks (45), RDS (104), and Redshift (26). The main pane displays a search bar with 'Search Metrics' and a placeholder 'Your CloudWatch metric summary has loaded. Total metrics: 1,014'. Below the search bar, there are four columns of metrics:

- Billing Metrics: 35
 - Total Estimated Charge: 1
 - By Service: 13
 - By Linked Account: 3
 - By Linked Account and Service: 18
- DynamoDB Metrics: 4
 - Table Metrics: 4
- EBS Metrics: 80
 - Per-Volume Metrics: 80
- EC2 Metrics: 272
 - Per-Instance Metrics: 181
 - By Auto Scaling Group: 56
 - By Image (AMI) Id: 14
 - Aggregated by Instance Type: 14
 - Across All Instances: 7
- ELB Metrics: 95
 - Per-LB Metrics: 29
 - Per LB, per AZ Metrics: 37
 - By Availability Zone: 20
 - Across All LBs: 9
- ElastiCache Metrics: 87
 - Cache Node Metrics: 87
- OpsWorks Metrics: 45
 - Instance Metrics: 15
 - Layer Metrics: 15
 - Stack Metrics: 15
- RDS Metrics: 104
 - Per-Database Metrics: 52
 - By Database Class: 26
 - By Database Engine: 13
 - Across All Databases: 13
- Redshift Metrics: 26
 - Node Metrics: 13
 - Aggregated by Cluster: 13

4. In the **CloudWatch Metrics by Category** pane, in the **Search Metrics** field, type a search term, metric name, service name, etc. and press enter.

For example, you can enter **volume** in the **Search Metrics** field, which returns all metrics with the word **volume** in their name.

The screenshot shows the CloudWatch Metrics console. On the left, there's a navigation pane with various services like Alarms, Billing, Metrics, and EBS. Under Metrics, there are sub-options like Selected Metrics, Billing, DynamoDB, EBS, EC2, ELB, ElastiCache, OpsWorks, RDS, Redshift, Route 53, SNS, and SQS. A red circle with the number 11 is on the Alarms icon. In the center, there's a search bar labeled 'Browse Metrics' with the query 'volume' entered. Below the search bar, it says 'Showing all results (80) for volume.' There are two buttons: 'Select All' and 'Clear'. The results table has two columns: 'VolumeId' and 'Metric Name'. The 'Metric Name' column contains 20 entries, each preceded by a checkbox. The first few entries are: 'vol-2ca75b6f', 'VolumeIdleTime', 'VolumeQueueLength', 'VolumeReadBytes', 'VolumeReadOps', 'VolumeTotalReadTime', 'VolumeTotalWriteTime', 'VolumeWriteBytes', 'VolumeWriteOps', 'VolumeIdleTime', 'VolumeQueueLength', 'VolumeReadBytes', 'VolumeReadOps', 'VolumeTotalReadTime', 'VolumeTotalWriteTime'. A red box highlights the 'VolumeIdleTime' entry. At the bottom right of the results table, there are buttons for 'Update Graph' and other visualization options.

Select and Deselect Metrics

You can select and deselect metrics in the CloudWatch console in many different ways. When you select metrics, they automatically appear in a graph in the details pane, so it's useful to know how to select or deselect metrics to graph the data you want.

To select or deselect metrics

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Metrics**.

Amazon CloudWatch Developer Guide

Select and Deselect Metrics

4. In the **CloudWatch Metrics by Category** pane, select a metrics category or in the **Search Metrics** field, type a search term, metric name, service name, and so on and press **Enter**.

For example, you can enter **volume** in the **Search Metrics** field, which returns all metrics with the word **volume** in their name.

5. Do one of the following:

- To select or deselect an individual metric, in the results pane, select the check box next to the resource name and metric.

EC2 > Per-Instance Metrics	
	Metric Name
<input checked="" type="checkbox"/>	i-14d5ac6c CPUUtilization
<input type="checkbox"/>	i-14d5ac6c DiskReadBytes

- To select all metrics in the list, in the results pane, at the top of the list, click **Select All**.

To deselect all metrics, in the results pane, at the top of the metrics list, click **Clear**.

EC2 > Per-Instance Metrics	
	Metric Name
<input checked="" type="checkbox"/>	i-14d5ac6c CPUUtilization
<input checked="" type="checkbox"/>	i-14d5ac6c DiskReadBytes
<input checked="" type="checkbox"/>	i-14d5ac6c DiskReadOps
<input checked="" type="checkbox"/>	i-14d5ac6c DiskWriteBytes
<input checked="" type="checkbox"/>	i-14d5ac6c StatusCheckFailed

- To list all resources that use a metric, in the results pane, in the **Metric Name** column, click a metric.

This is useful when you want to see all of these resources on the same graph. For more information, see [Graph a Metric Across Resources \(p. 53\)](#).

Select All | Clear

EC2 > Per-Instance Metrics

InstanceId	Metric Name
i-14d5ac6c	CPUUtilization
i-1697497c	CPUUtilization
i-1a384062	CPUUtilization
i-236bf44e	CPUUtilization
i-2b6bf446	CPUUtilization
i-2bf56152	CPUUtilization
i-33c2ea54	CPUUtilization

- To deselect all but one metric, in the results pane, in the metrics list, click the space between the resource type and the metric name of the metric you want to keep selected.

Select All | Clear

EC2 > Per-Instance Metrics

InstanceId	Metric Name
i-14d5ac6c	CPUUtilization
<input checked="" type="checkbox"/> i-1697497c	CPUUtilization
i-1a384062	CPUUtilization
i-236bf44e	CPUUtilization
i-2b6bf446	CPUUtilization
i-2bf56152	CPUUtilization

- To view a list of all selected metrics, in the navigation pane, under **Metrics**, click **Selected Metrics**.

The screenshot shows the CloudWatch Metrics console. On the left, there's a navigation pane with links like Dashboard, Alarms, ALARM (with 11 notifications), INSUFFICIENT (with 50 notifications), OK (with 33 notifications), Billing, Metrics, Selected Metrics (with 35 notifications), and other services like DynamoDB, EBS, EC2, ELB, and ElastiCache. The 'Metrics' link is highlighted with a red box. The main pane shows a search bar for 'CPUUtilization' and a table titled 'EC2 > Per-Instance Metrics'. The table lists seven instances with their CPUUtilization metric selected. A red box highlights the 'Selected Metrics' link in the navigation pane and the 'CPUUtilization' column header in the table.

InstanceId	Metric Name
i-14d5ac6c	CPUUtilization
i-1697497c	CPUUtilization
i-1a384062	CPUUtilization
i-236bf44e	CPUUtilization
i-2b6bf446	CPUUtilization
i-2bf56152	CPUUtilization
i-33c2ea54	CPUUtilization

Get Statistics for a Metric

This set of scenarios shows you how you can use the AWS Management Console, the `get-metric-statistics` command, or the `GetMetricStatistics` API to get a variety of statistics.

Note

Start and end times must be within the last 14 days.

Topics

- [Get Statistics for a Specific EC2 Instance \(p. 34\)](#)
- [Aggregating Statistics Across Instances \(p. 39\)](#)
- [Get Statistics Aggregated by Auto Scaling Group \(p. 44\)](#)
- [Get Statistics Aggregated by Image \(AMI\) ID \(p. 47\)](#)

Get Statistics for a Specific EC2 Instance

The following table describes the types of monitoring data available for your Amazon EC2 instances.

Monitoring Type	Description
Basic	Data is available automatically in 5-minute periods at no charge.
Detailed	Data is available in 1-minute periods at an additional cost. To get this level of data, you must specifically enable it for the instance. For the instances where you've enabled detailed monitoring, you can also get aggregated data across groups of similar instances. For information about pricing, go to the Amazon CloudWatch product page .

The following scenario walks you through how to use the AWS Management Console, the `get-metric-statistics` command, or the `GetMetricStatistics` API to determine the maximum CPU utilization of a specific EC2 instance. For more information about monitoring EC2 instances, see [Monitoring Your Instances with CloudWatch](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

Start and end times must be within the last 14 days.

For this example, we assume that you have an EC2 instance ID. You can get an active EC2 instance ID through the AWS Management Console.

AWS Management Console

To display the average CPU utilization for a specific instance

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).

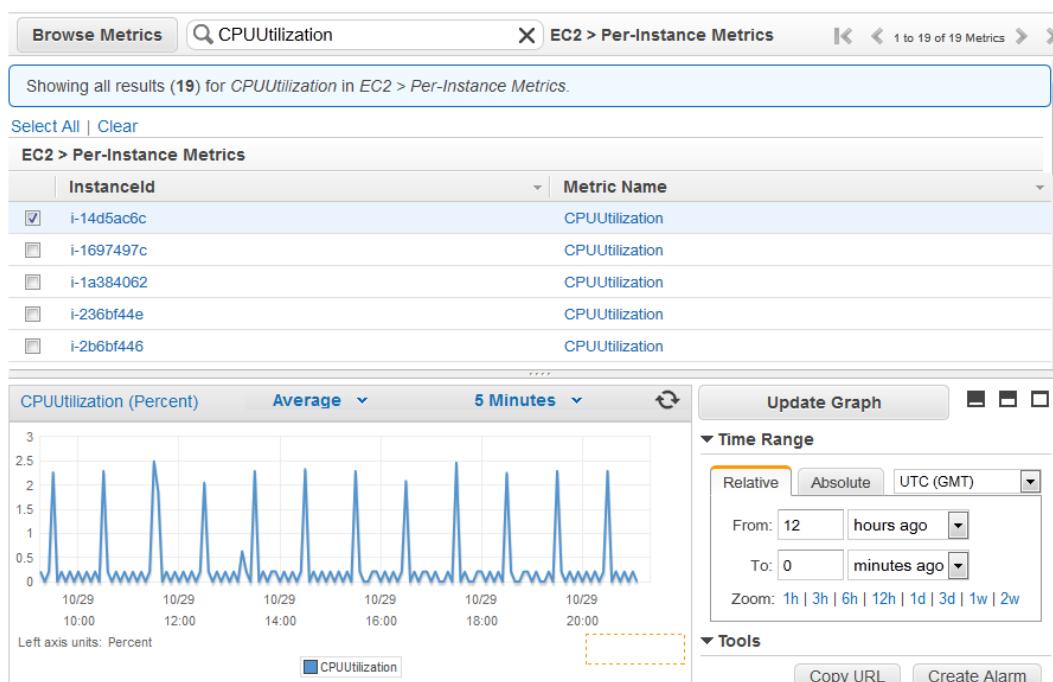


3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select **EC2: Metrics**.

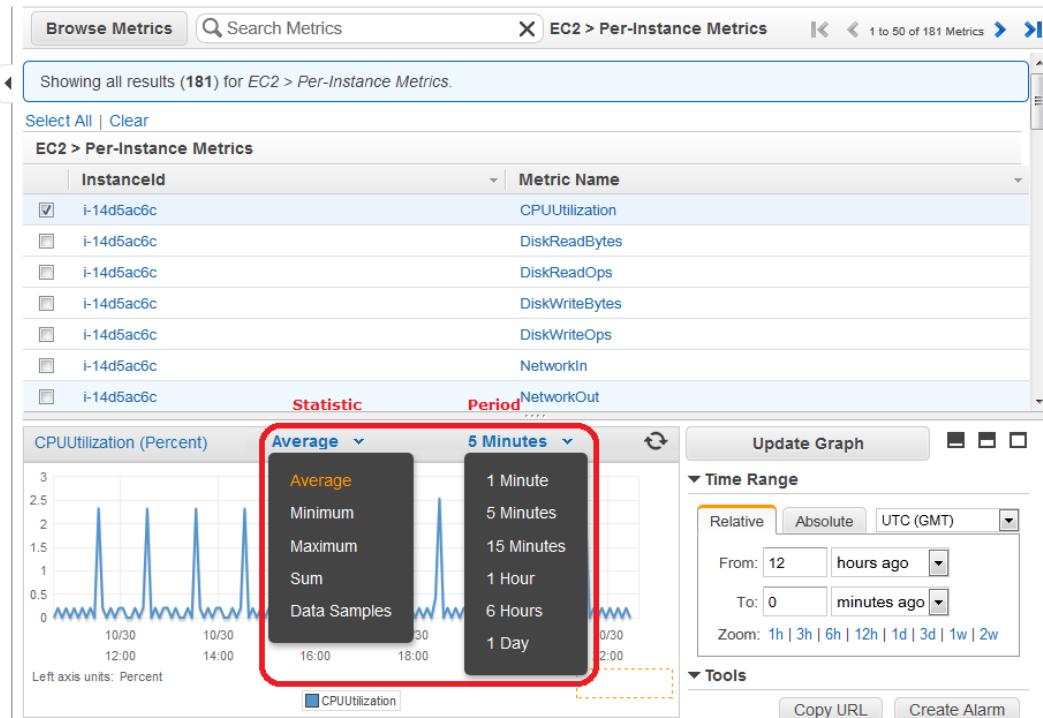
The metrics available for individual instances appear in the upper pane.

5. Select a row that contains **CPUUtilization** for a specific InstanceId.

A graph showing average CPUUtilization for a single instance appears in the details pane.



6. To change the **Statistic**, e.g., Average, for the metric, choose a different value from the pop-up list.



- To change the **Period**, e.g., 5 Minutes, to view data in more granular detail, choose a different value from the pop-up list.

Command Line Tools

To get the CPU utilization per EC2 instance

- Enter the `get-metric-statistics` command with the following parameters

```
Prompt>aws cloudwatch get-metric-statistics --metric-name CPUUtilization \
--start-time 2014-02-18T23:18:00 --end-time 2014-02-19T23:18:00 --period 3600 \
--namespace AWS/EC2 --statistics Maximum --dimensions Name=InstanceId,Value=<your-instance-id>
```

The AWS CLI returns the following:

```
{
    "Datapoints": [
        {
            "Timestamp": "2014-02-19T00:18:00Z",
            "Maximum": 0.33000000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T03:18:00Z",
            "Maximum": 99.67000000000002,
            "Unit": "Percent"
        }
    ]
}
```

```
        "Timestamp": "2014-02-19T07:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T12:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T02:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T01:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T17:18:00Z",
        "Maximum": 3.3900000000000001,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T13:18:00Z",
        "Maximum": 0.3300000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-18T23:18:00Z",
        "Maximum": 0.6700000000000004,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T06:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T11:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T10:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T19:18:00Z",
        "Maximum": 8.0,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-19T15:18:00Z",
        "Maximum": 0.3400000000000002,
        "Unit": "Percent"
    }
```

```

        },
        {
            "Timestamp": "2014-02-19T14:18:00Z",
            "Maximum": 0.3400000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T16:18:00Z",
            "Maximum": 0.3400000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T09:18:00Z",
            "Maximum": 0.3400000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T04:18:00Z",
            "Maximum": 2.0,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T08:18:00Z",
            "Maximum": 0.6800000000000005,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T05:18:00Z",
            "Maximum": 0.3300000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-19T18:18:00Z",
            "Maximum": 6.669999999999999,
            "Unit": "Percent"
        }
    ],
    "Label": "CPUUtilization"
}

```

The returned statistics are six-minute values for the requested two-day time interval. Each value represents the maximum CPU utilization percentage for a single EC2 instance.

Query API

To get the CPU utilization per hour for an EC2 instance for a 3-day range

- Call `GetMetricStatistics` with the following parameters:

- `MetricName = CPUUtilization`
- `Period = 3600`
- `Statistics list includes Maximum`
- `Dimensions (Name=InstanceId, Value="<your-instance-id>")`
- `Namespace = AWS/EC2`
- `StartTime = 2011-01-09T23:18:00`

- *EndTime* = `2011-01-12T23:18:00`

Aggregating Statistics Across Instances

Aggregate statistics are available for the instances that have detailed monitoring enabled. Instances that use basic monitoring are not included in the aggregates. In addition, Amazon CloudWatch does not aggregate data across Regions. Therefore, metrics are completely separate between Regions. Before you can get statistics aggregated across instances, you must enable detailed monitoring (at an additional charge), which provides data in 1-minute periods. This scenario shows you how to use detailed monitoring with either the AWS Management Console, the `GetMetricStatistics` API, or the `get-metric-statistics` command to get the average CPU usage for your EC2 instances. Because no dimension is specified, CloudWatch returns statistics for all dimensions in the `AWS/EC2` namespace. To get statistics for other metrics, see [Amazon CloudWatch Namespaces, Dimensions, and Metrics Reference \(p. 175\)](#).

Important

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

AWS Management Console

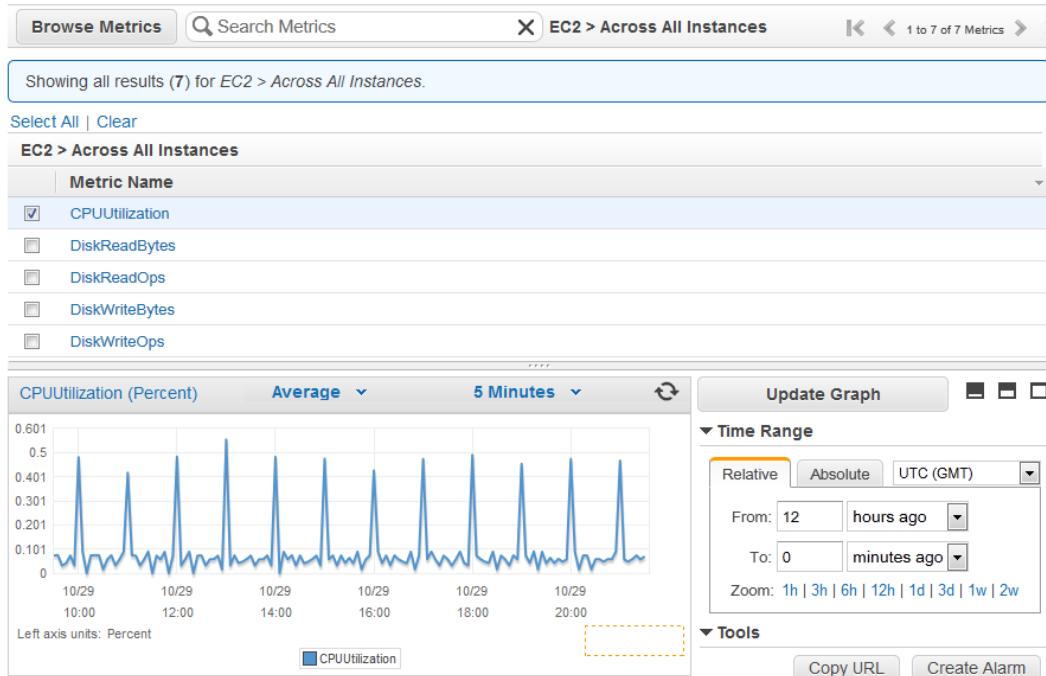
To display average CPU utilization for your Amazon EC2 instances

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).

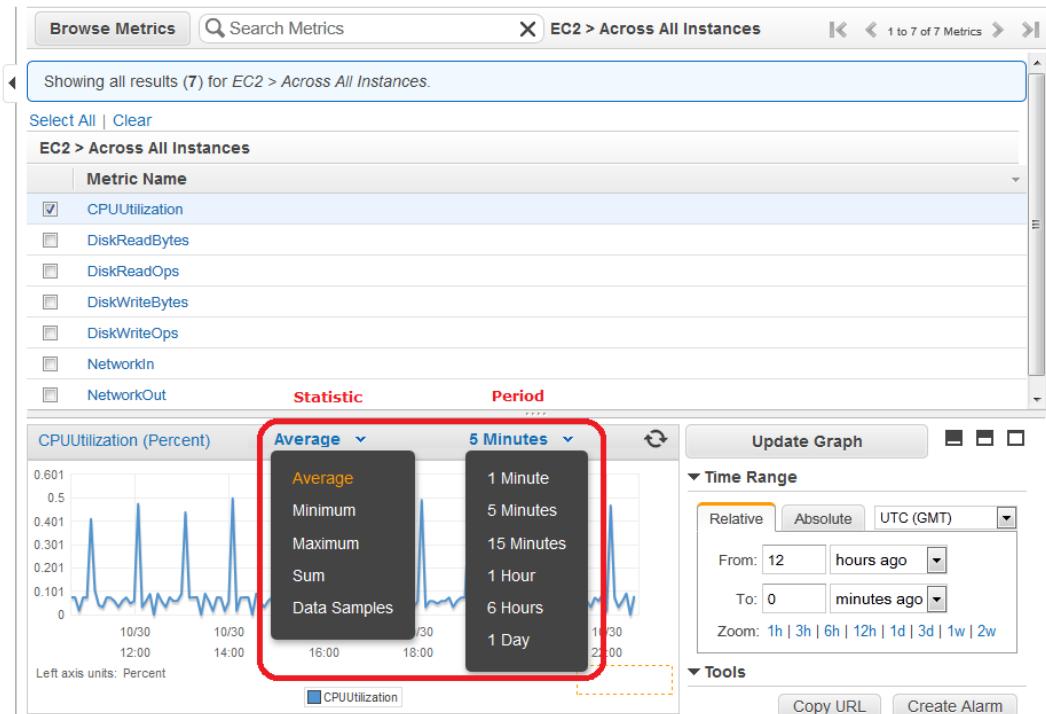


3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, under **EC2 Metrics**, select **Across All Instances**.
The metrics available across all instances are displayed in the upper pane.
5. In the upper pane, select the row that contains **CPUUtilization**.

A graph showing CPUUtilization for your EC2 instances is displayed in the details pane.



- To change the **Statistic**, e.g., Average, for the metric, choose a different value from the pop-up list.



- To change the **Period**, e.g., 5 Minutes, to view data in more granular detail, choose a different value from the pop-up list.

Command Line Tools

To get average CPU utilization across your Amazon EC2 instances

- Enter the `get-metric-statistics` command with the following parameters:

```
Prompt>aws cloudwatch get-metric-statistics --metric-name CPUUtilization -  
-start-time 2014-02-11T23:18:00 --end-time 2014-02-12T23:18:00 --period 3600  
--namespace AWS/EC2 --statistics "Average" "SampleCount"
```

The AWS CLI returns the following:

```
{  
    "Datapoints": [  
        {  
            "SampleCount": 238.0,  
            "Timestamp": "2014-02-12T07:18:00Z",  
            "Average": 0.038235294117647062,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 240.0,  
            "Timestamp": "2014-02-12T09:18:00Z",  
            "Average": 0.1667083333333332,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 238.0,  
            "Timestamp": "2014-02-11T23:18:00Z",  
            "Average": 0.041596638655462197,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 240.0,  
            "Timestamp": "2014-02-12T16:18:00Z",  
            "Average": 0.03945833333333345,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 239.0,  
            "Timestamp": "2014-02-12T21:18:00Z",  
            "Average": 0.041255230125523033,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 240.0,  
            "Timestamp": "2014-02-12T01:18:00Z",  
            "Average": 0.04458333333333336,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 239.0,  
            "Timestamp": "2014-02-12T18:18:00Z",  
            "Average": 0.043054393305439344,  
            "Unit": "Percent"  
        },  
        {  
            "SampleCount": 240.0,  
            "Timestamp": "2014-02-12T23:18:00Z",  
            "Average": 0.04458333333333336,  
            "Unit": "Percent"  
        }]
```

```
{  
    "SampleCount": 240.0,  
    "Timestamp": "2014-02-12T13:18:00Z",  
    "Average": 0.03945833333333345,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 238.0,  
    "Timestamp": "2014-02-12T15:18:00Z",  
    "Average": 0.041260504201680689,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 240.0,  
    "Timestamp": "2014-02-12T19:18:00Z",  
    "Average": 0.03766666666666668,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 240.0,  
    "Timestamp": "2014-02-12T06:18:00Z",  
    "Average": 0.03754166666666675,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 240.0,  
    "Timestamp": "2014-02-12T20:18:00Z",  
    "Average": 0.0393333333333338,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 240.0,  
    "Timestamp": "2014-02-12T08:18:00Z",  
    "Average": 0.03925000000000014,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 239.0,  
    "Timestamp": "2014-02-12T03:18:00Z",  
    "Average": 0.037740585774058588,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 240.0,  
    "Timestamp": "2014-02-12T11:18:00Z",  
    "Average": 0.03950000000000007,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 238.0,  
    "Timestamp": "2014-02-12T02:18:00Z",  
    "Average": 0.039789915966386563,  
    "Unit": "Percent"  
},  
{  
    "SampleCount": 238.0,  
    "Timestamp": "2014-02-12T22:18:00Z",  
    "Average": 0.039705882352941181,  
}
```

```
        "Unit": "Percent"
    },
{
    "SampleCount": 240.0,
    "Timestamp": "2014-02-12T14:18:00Z",
    "Average": 0.0824583333333328,
    "Unit": "Percent"
},
{
    "SampleCount": 240.0,
    "Timestamp": "2014-02-12T05:18:00Z",
    "Average": 0.04287500000000001,
    "Unit": "Percent"
},
{
    "SampleCount": 240.0,
    "Timestamp": "2014-02-12T17:18:00Z",
    "Average": 0.0394583333333345,
    "Unit": "Percent"
},
{
    "SampleCount": 240.0,
    "Timestamp": "2014-02-12T10:18:00Z",
    "Average": 0.0834166666666667,
    "Unit": "Percent"
},
{
    "SampleCount": 236.0,
    "Timestamp": "2014-02-12T00:18:00Z",
    "Average": 0.036567796610169498,
    "Unit": "Percent"
},
{
    "SampleCount": 240.0,
    "Timestamp": "2014-02-12T12:18:00Z",
    "Average": 0.03954166666666676,
    "Unit": "Percent"
},
{
    "SampleCount": 240.0,
    "Timestamp": "2014-02-12T04:18:00Z",
    "Average": 0.04300000000000003,
    "Unit": "Percent"
}
],
"Label": "CPUUtilization"
}
```

Query API

To get average CPU utilization for your Amazon EC2 instances

- Call `GetMetricStatistics` with the following parameters:
 - `MetricName = CPUUtilization`

- *Statistics* list includes Average
- *Namespace* = AWS/EC2
- *StartTime* = **2011-01-10T23:18:00**
- *EndTime* = **2011-01-12T23:18:00**
- *Period* = 360

The returned statistics are six-minute values for the two-day interval.

Get Statistics Aggregated by Auto Scaling Group

This scenario shows you how to use the AWS Management Console, the `get-metric-statistics` command, or the `GetMetricStatistics` API with the `DiskWriteBytes` metric to retrieve the total bytes written to disk for one Auto Scaling group. The total is computed for one-minute periods for a 24-hour interval across all EC2 instances in the specified `AutoScalingGroupName`.

Note

Start and end times must be within the last 14 days.

We assume for this example that an EC2 application is running and has an Auto Scaling group named `test-group-1`.

AWS Management Console

To display total DiskWriteBytes for an autoscaled EC2 application

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



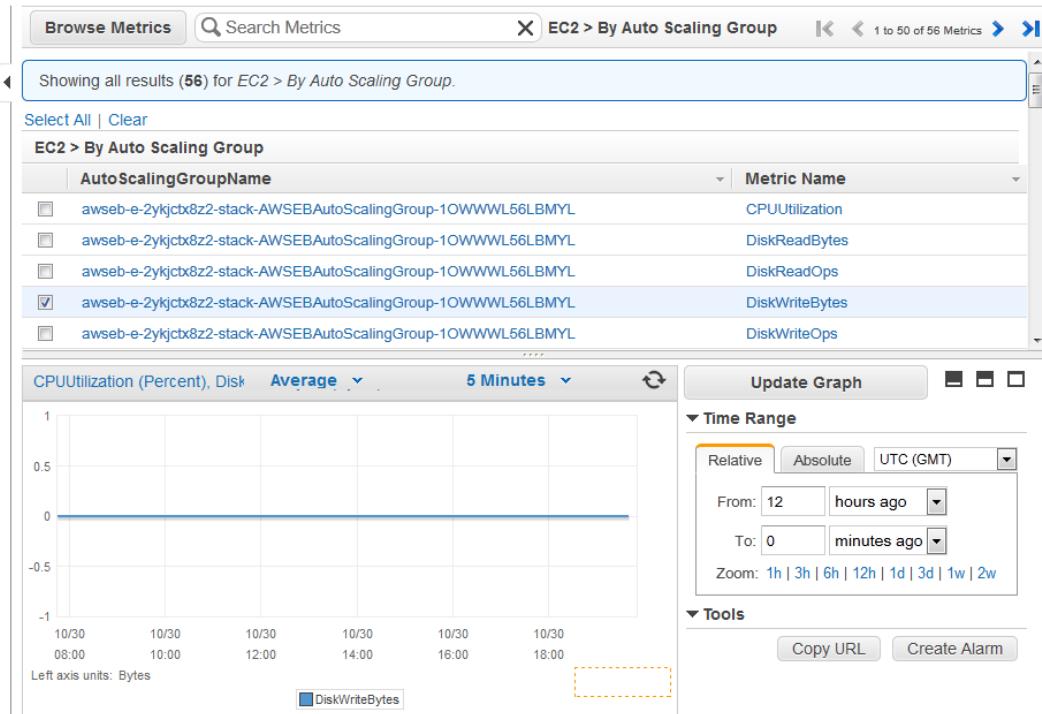
3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, under **EC2 Metrics**, select **By Auto Scaling Group**.

The metrics available for Auto Scaling groups are displayed in the upper pane.

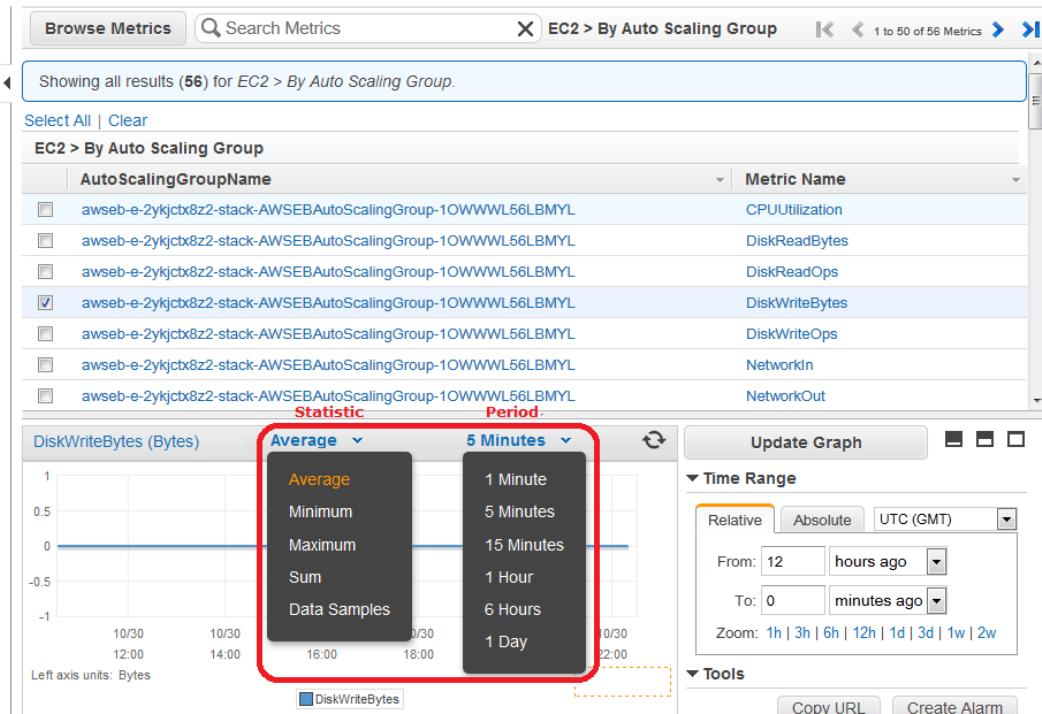
5. Select the row that contains **DiskWriteBytes**.

A graph showing `DiskWriteBytes` for all EC2 instances appears in the details pane.

Amazon CloudWatch Developer Guide
Get Statistics Aggregated by Auto Scaling Group



6. To change the **Statistic**, e.g., Average, for the metric, choose a different value from the pop-up list.



7. To change the **Period**, e.g., 5 Minutes, to view data in more granular detail, choose a different value from the pop-up list.

Command Line Tools

To get total DiskWriteBytes for an autoscaled EC2 application

- Enter the `get-metric-statistics` command with the following parameters.

```
Prompt>aws cloudwatch get-metric-statistics --metric-name DiskWriteBytes -  
--start-time 2014-02-16T23:18:00 --end-time 2014-02-18T23:18:00 --period 360  
--namespace AWS/EC2 --statistics "Sum" "SampleCount" --dimensions  
Name=AutoScalingGroupName,Value=test-group-1
```

The AWS CLI returns the following:

```
{  
    "Datapoints": [  
        {  
            "SampleCount": 18.0,  
            "Timestamp": "2014-02-19T21:36:00Z",  
            "Sum": 0.0,  
            "Unit": "Bytes"  
        },  
        {  
            "SampleCount": 5.0,  
            "Timestamp": "2014-02-19T21:42:00Z",  
            "Sum": 0.0,  
            "Unit": "Bytes"  
        }  
    ],  
    "Label": "DiskWriteBytes"  
}
```

Query API

To get total DiskWriteBytes for an autoscaled EC2 application

- Call `GetMetricStatistics` with the following parameters:

- MetricName* = `DiskWriteBytes`
- Period* = 60
- Statistics* list includes `Sum`
- Unit* = `Bytes`
- Dimensions* (`Name=AutoScalingGroupName,Value=test-group-1`)
- Namespace* = `AWS/EC2`
- StartTime* = `2011-01-10T23:18:00`
- EndTime* = `2011-01-11T23:18:00`

The statistics returned are one-minute totals for bytes written for the entire Auto Scaling group over the 24-hour interval.

Get Statistics Aggregated by Image (AMI) ID

This scenario shows you how to use the AWS Management Console, the `get-metric-statistics` command, or the `GetMetricStatistics` API to determine average CPU utilization for all instances that match a given image ID. The average is over 60-second time intervals for a one-day period.

Note

Start and end times must be within the last 14 days.

In this scenario the EC2 instances are running an image ID of `ami-c5e40dac`.

AWS Management Console

To display the average CPU utilization for an image ID

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).

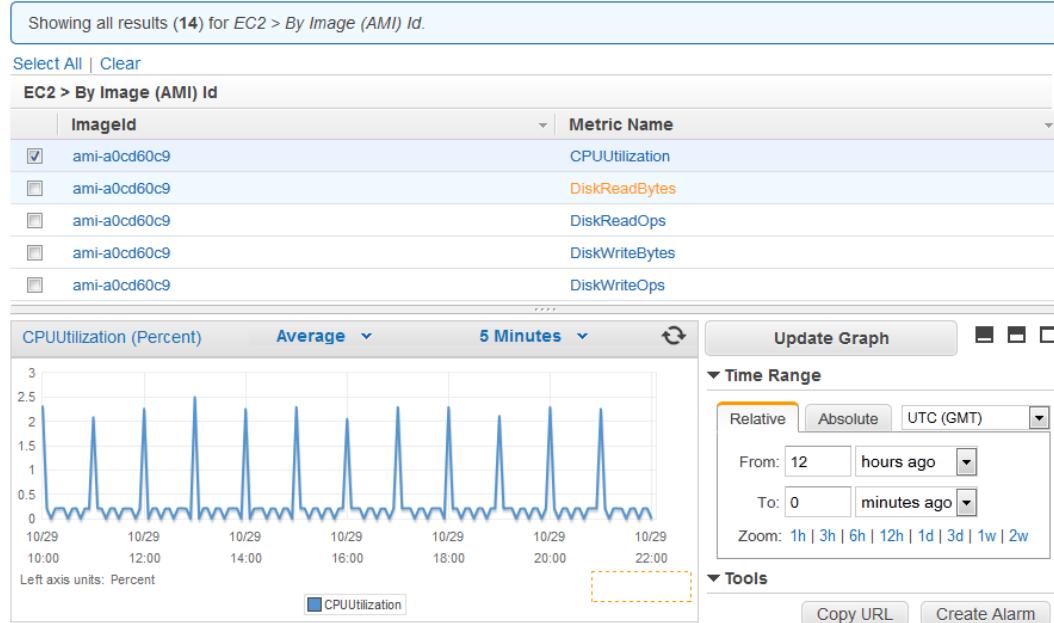


3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, under **EC2 Metrics**, select **By Image (AMI) Id**.
The metrics available for image IDs appear in the upper pane.
5. Select a row that contains **CPUUtilization** and an image ID.

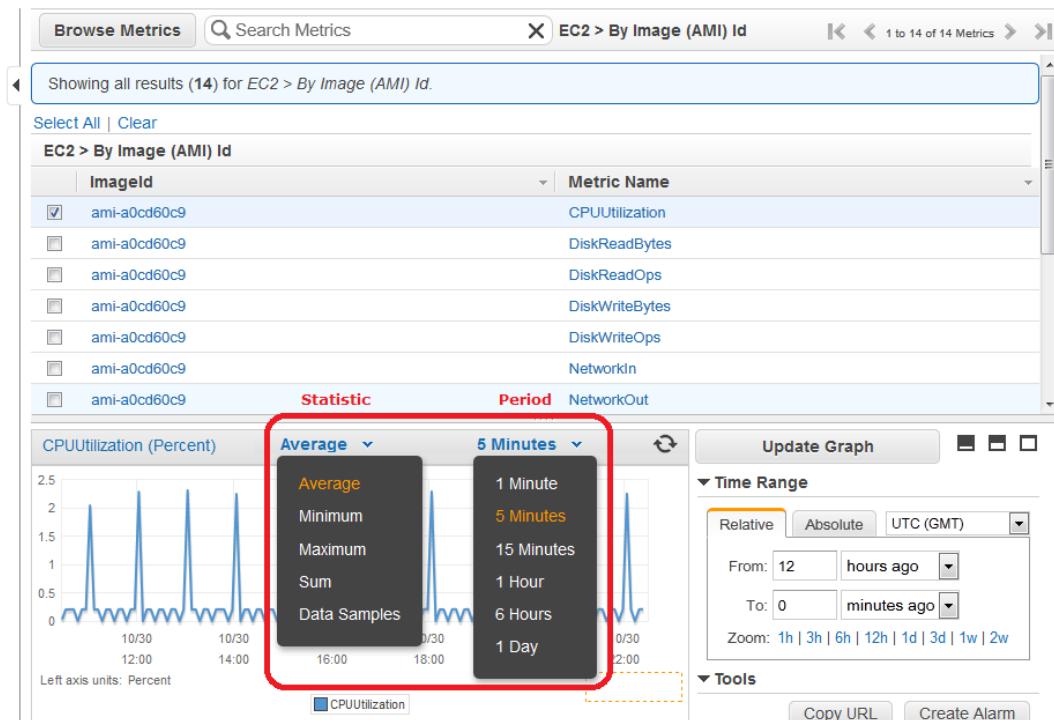
A graph showing average `CPUUtilization` for all EC2 instances based on the `ami-c5e40dac` image ID appears in the details pane.

Amazon CloudWatch Developer Guide

Get Statistics Aggregated by Image (AMI) ID



6. To change the **Statistic**, e.g., Average, for the metric, choose a different value from the pop-up list.



7. To change the **Period**, e.g., 5 Minutes, to view data in more granular detail, choose a different value from the pop-up list.

Command Line Tools

To get the average CPU utilization for an image ID

- Enter the `get-metric-statistics` command as in the following example.

```
Prompt>aws cloudwatch get-metric-statistics --metric-name CPUUtilization -  
-start-time 2014-02-10T00:00:00 --end-time 2014-02-11T00:00:00 --period 3600  
--statistics Average --namespace AWS/EC2 --dimensions Name="Im  
ageId",Value=ami-3c47a355
```

The AWS CLI returns the following:

```
{  
    "Datapoints": [  
        {  
            "Timestamp": "2014-02-10T07:00:00Z",  
            "Average": 0.04100000000000009,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T14:00:00Z",  
            "Average": 0.079579831932773085,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T06:00:00Z",  
            "Average": 0.03600000000000011,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T13:00:00Z",  
            "Average": 0.037625000000000013,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T18:00:00Z",  
            "Average": 0.04275000000000003,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T21:00:00Z",  
            "Average": 0.039705882352941188,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T20:00:00Z",  
            "Average": 0.03937500000000007,  
            "Unit": "Percent"  
        },  
        {  
            "Timestamp": "2014-02-10T02:00:00Z",  
            "Average": 0.0410416666666671,  
            "Unit": "Percent"  
        }  
    ]  
}
```

```
        "Timestamp": "2014-02-10T01:00:00Z",
        "Average": 0.04108333333333354,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T23:00:00Z",
        "Average": 0.038016877637130804,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T15:00:00Z",
        "Average": 0.03766666666666668,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T12:00:00Z",
        "Average": 0.03929166666666676,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T03:00:00Z",
        "Average": 0.03600000000000004,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T04:00:00Z",
        "Average": 0.04266666666666672,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T19:00:00Z",
        "Average": 0.038305084745762719,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T22:00:00Z",
        "Average": 0.03929166666666676,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T09:00:00Z",
        "Average": 0.17126050420168065,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T08:00:00Z",
        "Average": 0.04116666666666678,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T11:00:00Z",
        "Average": 0.08237499999999962,
        "Unit": "Percent"
    },
    {
        "Timestamp": "2014-02-10T17:00:00Z",
        "Average": 0.03762500000000013,
        "Unit": "Percent"
    }
```

```
        } ,
        {
            "Timestamp": "2014-02-10T10:00:00Z",
            "Average": 0.03945833333333345,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-10T05:00:00Z",
            "Average": 0.03925000000000007,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-10T00:00:00Z",
            "Average": 0.03762500000000013,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-02-10T16:00:00Z",
            "Average": 0.041512605042016815,
            "Unit": "Percent"
        }
    ],
    "Label": "CPUUtilization"
}
```

The operation returns statistics that are one-minute values for the one-day interval. Each value represents an average CPU utilization percentage for EC2 instances running the specified machine image.

Query API

To get the average CPU utilization for an image ID

- Call `GetMetricStatistics` with the following parameters:
 - `MetricName` = `CPUUtilization`
 - `Period` = `60`
 - `Statistics` list includes `Average`
 - `Dimensions` (`Name= ImageId, Value= ami-c5e40dac`)
 - `Namespace` = `AWS/EC2`
 - `StartTime` = `2011-01-10T00:00:00`
 - `EndTime` = `2011-01-11T00:00:00`

Graph Metrics

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

Topics

- [Graph a Metric \(p. 52\)](#)
- [Graph a Metric Across Resources \(p. 53\)](#)

- [Graph Several Metrics \(p. 55\)](#)
- [Modify the Date and Time on a Graph \(p. 56\)](#)
- [Modify the Statistic for a Graph \(p. 57\)](#)
- [Modify the Period for a Graph \(p. 58\)](#)
- [Modify a Graph's Title \(p. 59\)](#)
- [Create an Alarm from a Metric on a Graph \(p. 61\)](#)
- [Zoom in to a Graph \(p. 62\)](#)
- [Move Backwards in Time on a Graph \(p. 63\)](#)
- [Move Forwards in Time on a Graph \(p. 64\)](#)
- [Jump to "Now" on a Graph \(p. 65\)](#)
- [Switch the Y-Axis for a Metric \(p. 66\)](#)
- [Save a Graph \(p. 67\)](#)

Graph a Metric

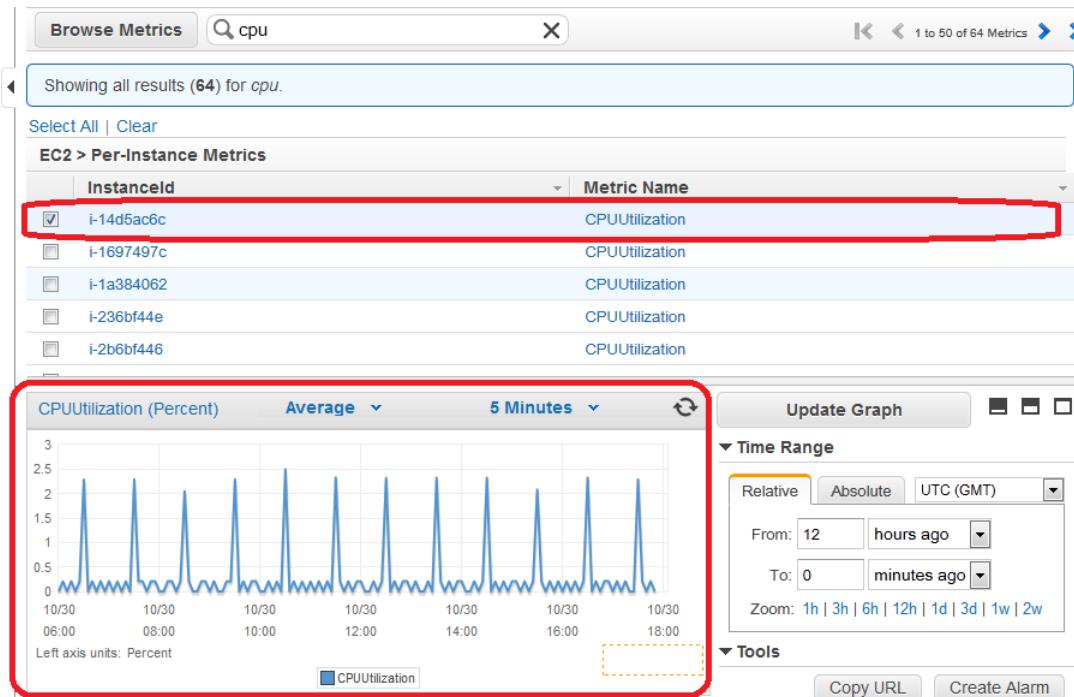
You can select a metric and create a graph of the data in CloudWatch. For example, you can select the CPUUtilization metric for an Amazon EC2 instance and display a graph of CPU usage over time for that instance.

To graph a metric

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, use the **Search Metrics** field and categories to find a metric by metric name, AWS resource, or other metadata.
5. Use the scroll bar and next and previous arrows above the metrics list to page through the full list of metrics
6. Select the metric to view, for example, CPUUtilization. A graph appears in the details pane.



- To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Graph a Metric Across Resources

You can graph a metric across all resources to see everything on one graph. For example, you can graph the CPUUtilization metric for all Amazon EC2 instances on one graph.

To graph a metric across resources

- Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
- If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



- In the navigation pane, click **Metrics**.

Amazon CloudWatch Developer Guide

Graph a Metric Across Resources

The screenshot shows the CloudWatch Metrics by Category pane. On the left, there's a sidebar with navigation links like Dashboard, Alarms, Metrics (selected), Billing, DynamoDB, EBS, EC2, ELB, ElastiCache, OpsWorks, RDS, Redshift, Route 53, SNS, and SQS. The Metrics section has three status indicators: ALARM (10), INSUFFICIENT (48), and OK (29). The main pane displays a grid of metric categories with their counts:

Billing Metrics	DynamoDB Metrics	EBS Metrics
35	4	80
Total Estimated Charge: 1 By Service: 13 By Linked Account: 3 By Linked Account and Service: 18		Per-Volume Metrics: 80
EC2 Metrics	ELB Metrics	ElastiCache Metrics
272	95	87
Per-Instance Metrics: 181 By Auto Scaling Group: 56 By Image (AMI) Id: 14 Aggregated by Instance Type: 14 Across All Instances: 7	Per-LB Metrics: 29 Per LB, per AZ Metrics: 37 By Availability Zone: 20 Across All LBs: 9	Cache Node Metrics: 87
OpsWorks Metrics	RDS Metrics	Redshift Metrics
45	104	26
Instance Metrics: 15 Layer Metrics: 15 Stack Metrics: 15	Per-Database Metrics: 52 By Database Class: 26 By Database Engine: 13 Across All Databases: 13	Node Metrics: 13 Aggregated by Cluster: 13

- In the **CloudWatch Metrics by Category** pane, select a metric category. For example, under **EC2 Metrics**, select **Per-Instance Metrics**.

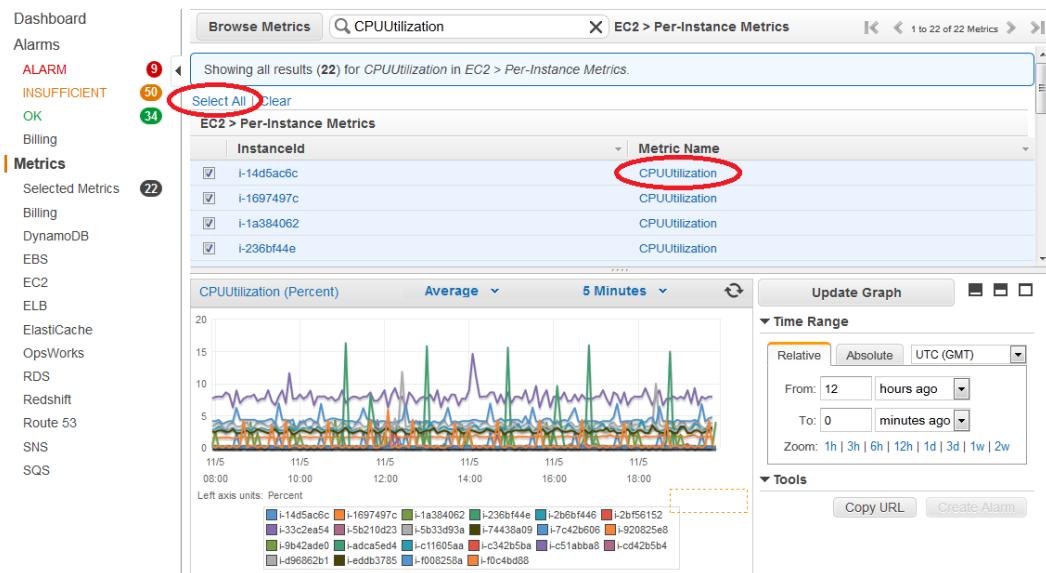
The screenshot shows the EC2 > Per-Instance Metrics list pane. At the top, it says "Showing all results (181) for EC2 > Per-Instance Metrics." Below is a table with columns "InstanceId" and "Metric Name". The table lists five instances (i-14d5ac6c) with their corresponding CPUUtilization metrics.

InstanceId	Metric Name
i-14d5ac6c	CPUUtilization
i-14d5ac6c	DiskReadBytes
i-14d5ac6c	DiskReadOps
i-14d5ac6c	DiskWriteBytes
i-14d5ac6c	DiskWriteOps

Below the table, there's a large upward arrow pointing to the text "Select a metric above to view graph". Further down, it says "Click a checkbox to select a metric" and "Click on text to add to search". To the right, there's a "Time Range" section with dropdowns for "From" (12 hours ago), "To" (0 minutes ago), and "Zoom" (1h, 3h, 6h, 12h, 1d, 3d, 1w, 2w). There's also an "Update Graph" button.

- In the metric list, in the **Metric Name** column, click a metric. For example **CPUUtilization**.
- At the top of the metric list, click **Select All**.

The graph shows all data for all occurrences of the selected metric. In the example below, CPUUtilization for all Amazon EC2 instances is shown.



- To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Graph Several Metrics

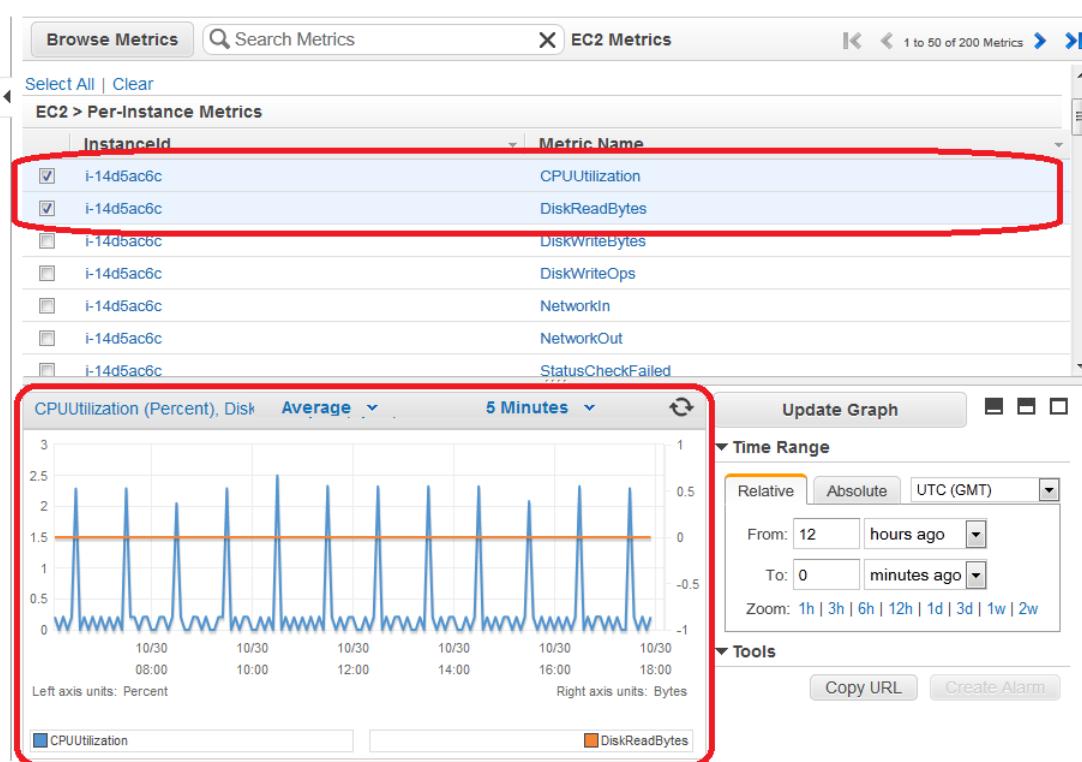
You can graph several metrics over time on the same graph. For example, you can graph CPUUtilization and DiskReadBytes for an Amazon EC2 instance and show them together on the same graph.

To graph several metrics

- Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
- If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



- In the navigation pane, click **Metrics**.
- In the **CloudWatch Metrics by Category** pane, use the **Search Metrics** field and categories to find a metric by metric name, AWS resource, or other metadata.
- Select the check box next to each metric you want to graph. You can add additional metrics by selecting their check boxes. A line appears on the graph for each check box you select.



6. To clear your selections and view data for a single metric, click the metric name in the list.
7. To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Modify the Date and Time on a Graph

You can change the date and time on a graph to view data at different points in time.

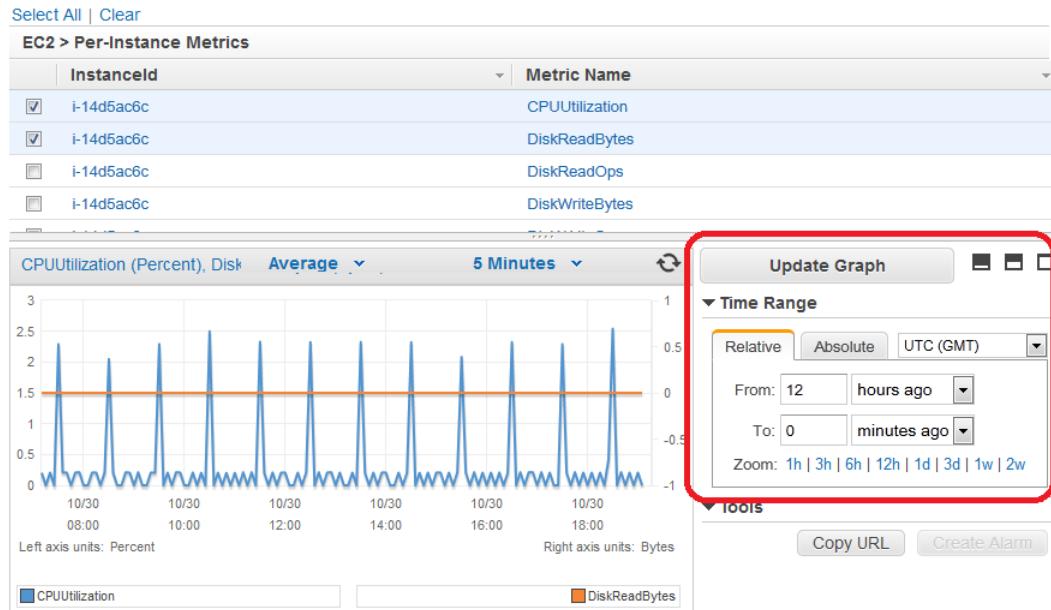
To modify the date and time on a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.

4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. In the details pane, in the **Time Range** section, select a new date range using the **From** and **To** fields.



7. Click **Update Graph** or the refresh (circular arrows) button to update the graph with the new date range.
8. To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Modify the Statistic for a Graph

CloudWatch supports several different statistics on metrics: **Average**, **Minimum**, **Maximum**, **Sum** & **Samples**. For more information, see [Statistics \(p. 6\)](#).

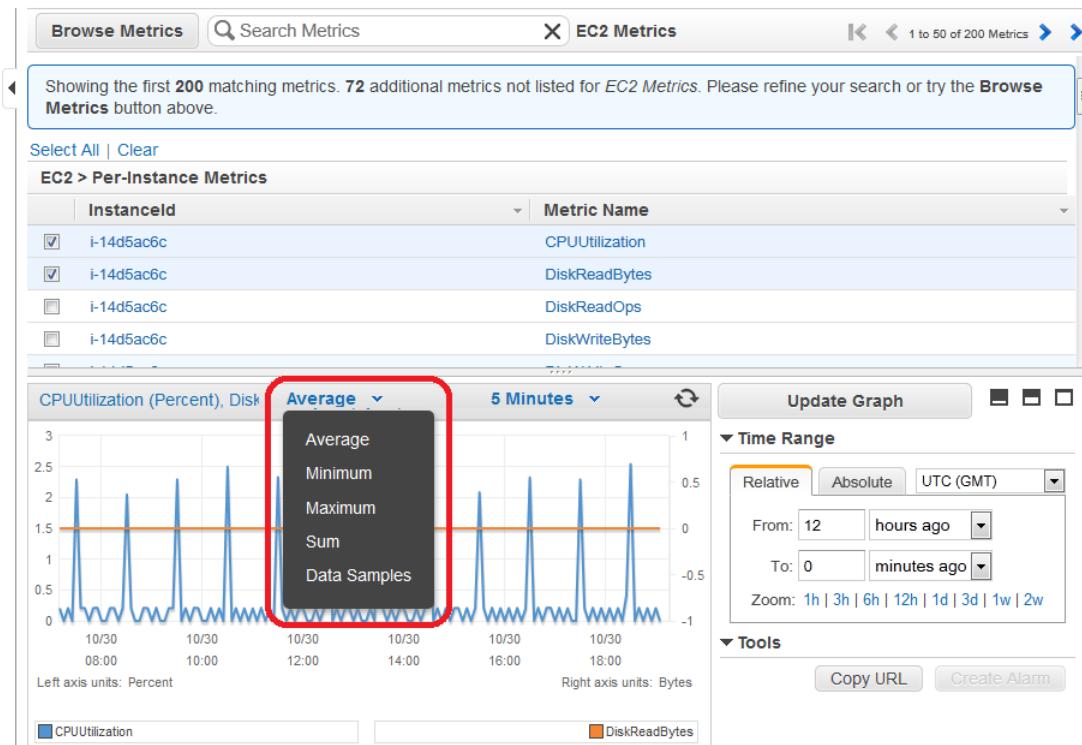
To modify the statistic for a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. Next to the graph's title, click the **Statistic** drop down list and then select a statistic, for example, **Maximum**.

The graph updates with the new selection.



7. To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Modify the Period for a Graph

CloudWatch enables you to view your data at different granularities. You can view your data using **Period** of *1 minute*, giving you a very detailed view. This is very useful when troubleshooting, when viewing narrow bands of time (e.g. 1 hour), and when performing other activities that require the most precise graphing of time periods. You can also view your data using **Period** of *1 hour*, giving you a less detailed view. This is very useful when viewing wider bands of time (e.g. 3 days), and allows you to see trends over time. For more information, see [Periods \(p. 7\)](#).

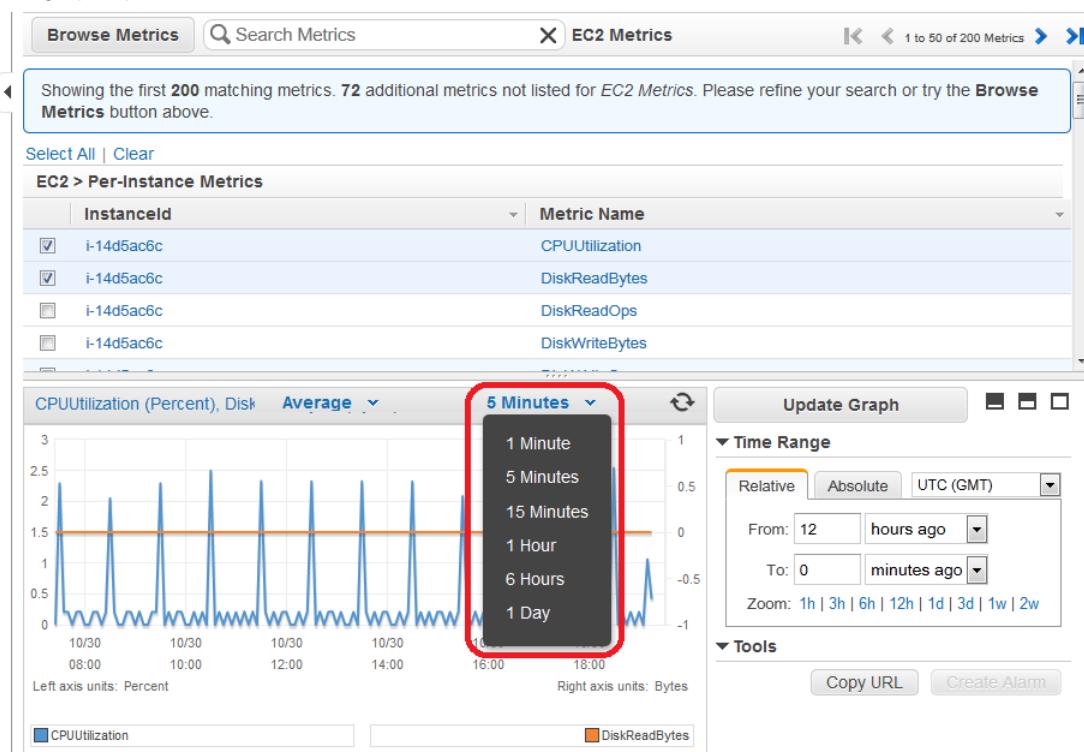
To modify the period for a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. Click the **Period** drop down list and then select a new period, for example, **5 Minutes**.

The graph updates with the new selection.



7. To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Modify a Graph's Title

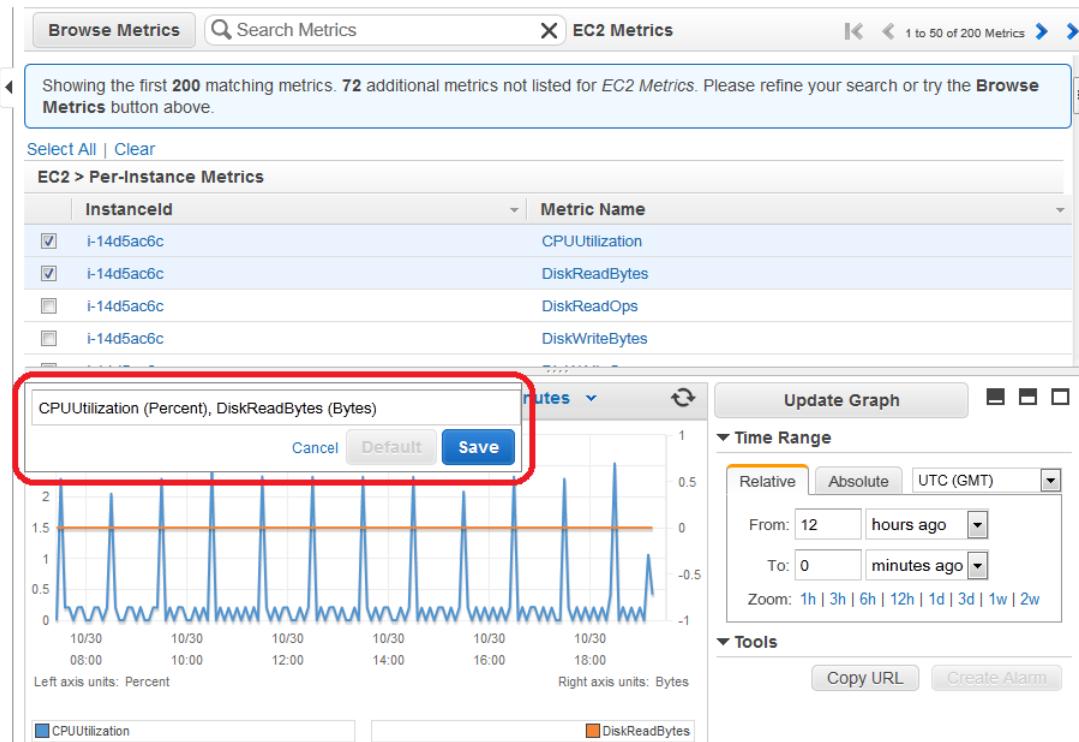
CloudWatch provides a default title for any graph you create. You can edit the title and change it if you want.

To modify a graph's title

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. In the details pane, click the title to edit it.
7. In the pop-up box, enter a new title, and then click **Save** to update the graph's title.



Create an Alarm from a Metric on a Graph

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

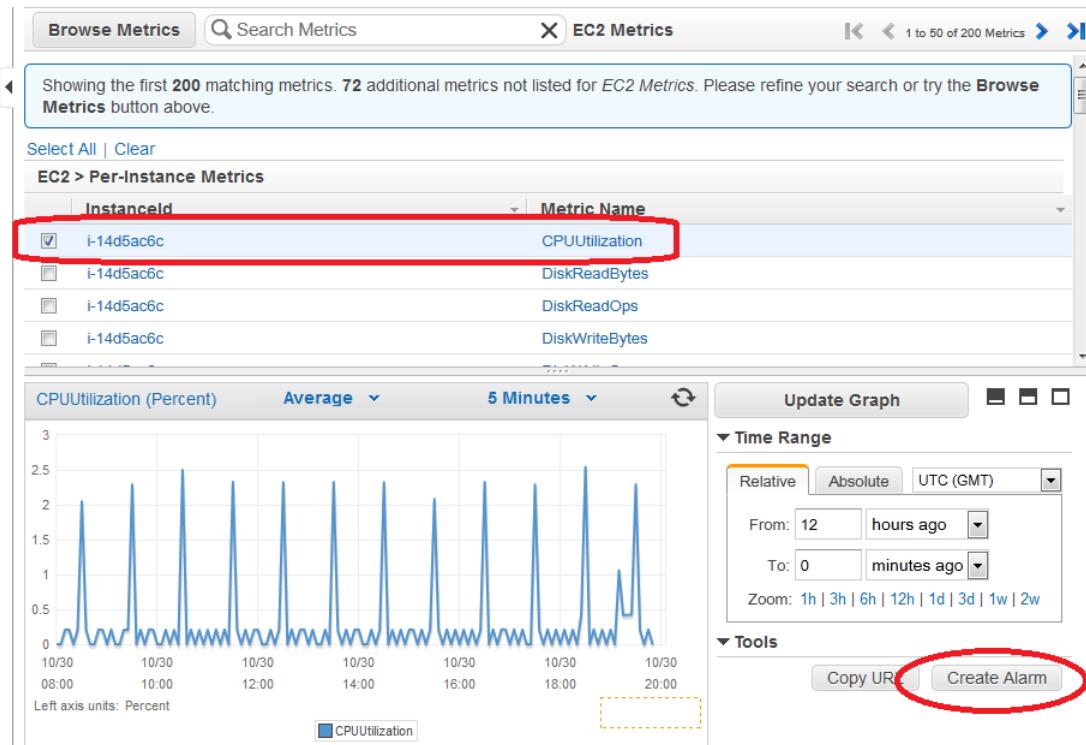
To create an alarm from a metric on a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box next to the metric for which you want to create an alarm.
6. In the details pane, under **Tools**, click **Create Alarm**, and then complete the alarm fields.

For more information about how to create an alarm, see [Creating Amazon CloudWatch Alarms \(p. 71\)](#).



Zoom in to a Graph

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

To zoom in to a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

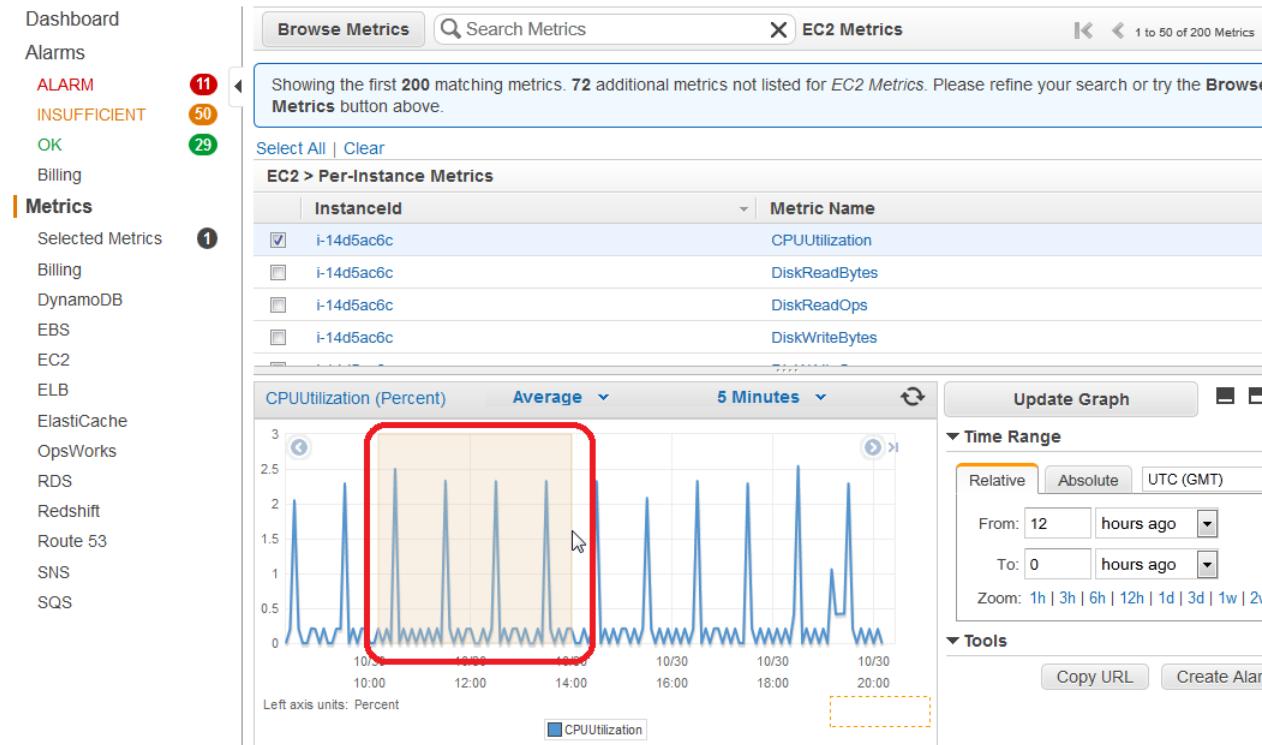


3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. In the details pane, click and drag on the graph area, and then release your mouse button.

Amazon CloudWatch Developer Guide

Move Backwards in Time on a Graph

The graph updates with the new selection.



7. To save this graph and access it later, in the details pane, under **Tools**, click **Copy URL**, and then in the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Move Backwards in Time on a Graph

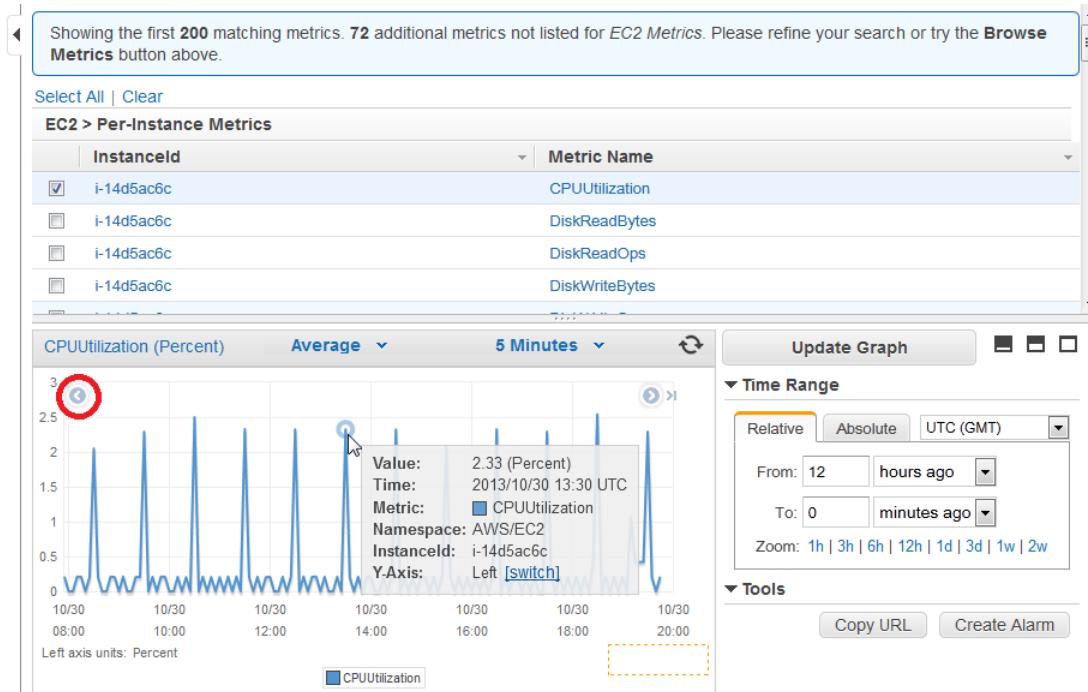
To move backwards in time on a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the [Amazon Web Services General Reference](#).



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.

6. In the details pane, hover over an empty space in the graph area and the time controls (left and right arrow buttons) appear at the left and right edges of the graph.
7. Click the left arrow. The graph updates with the new selection.



Move Forwards in Time on a Graph

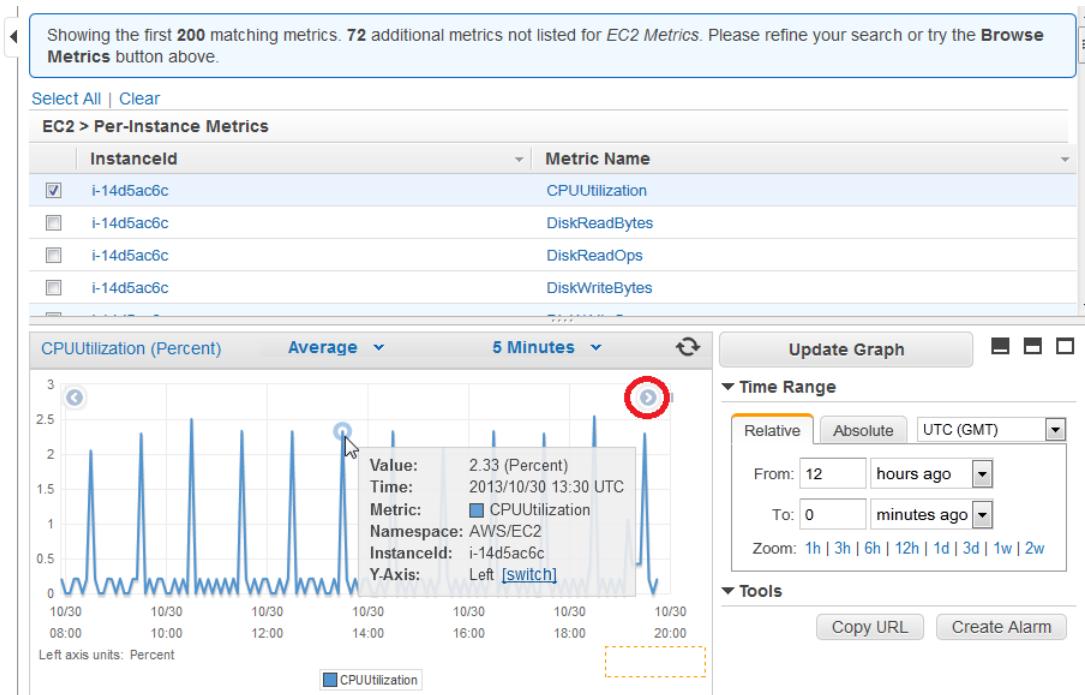
To move forwards in time on a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. In the details pane, hover over an empty space in the graph area and the time controls (left and right arrow buttons) appear at the left and right edges of the graph.

7. Click the right arrow. The graph updates with the new selection.



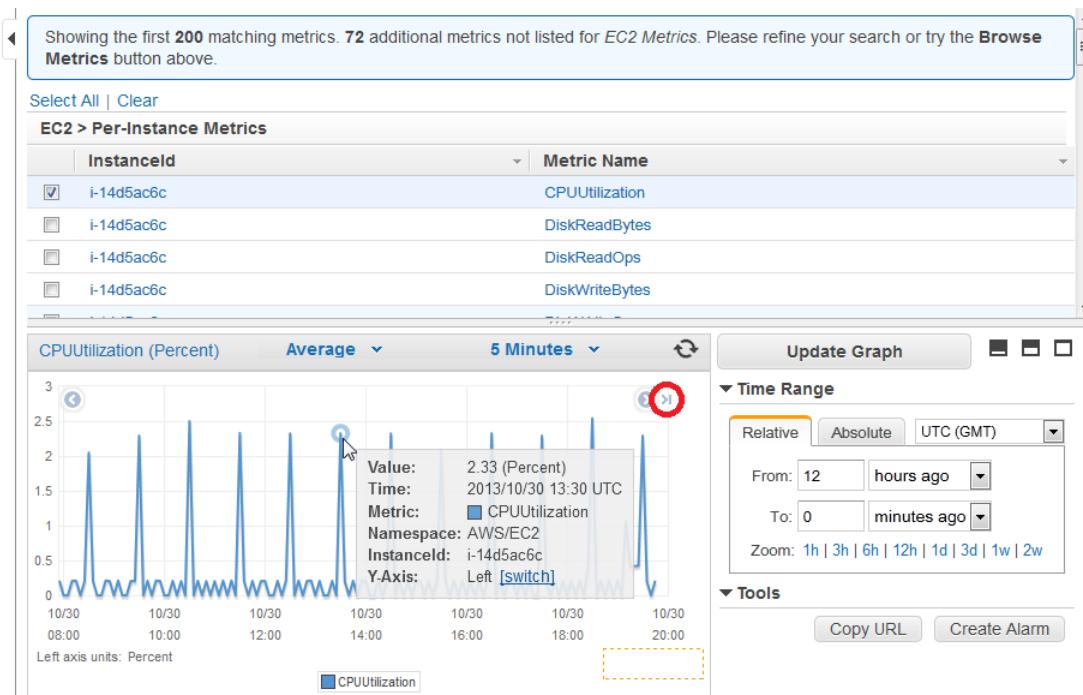
Jump to "Now" on a Graph

To jump to "now" on a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. In the details pane, hover over an empty space in the graph area and the time controls (left and right arrow buttons) appear at the left and right edges of the graph.
7. Click the right arrow with the vertical line. The graph updates with the new selection.



Switch the Y-Axis for a Metric

You can display multiple metrics on a single graph using two different Y-axes. This is particularly useful for metrics that have different units or that differ greatly in their range of values.

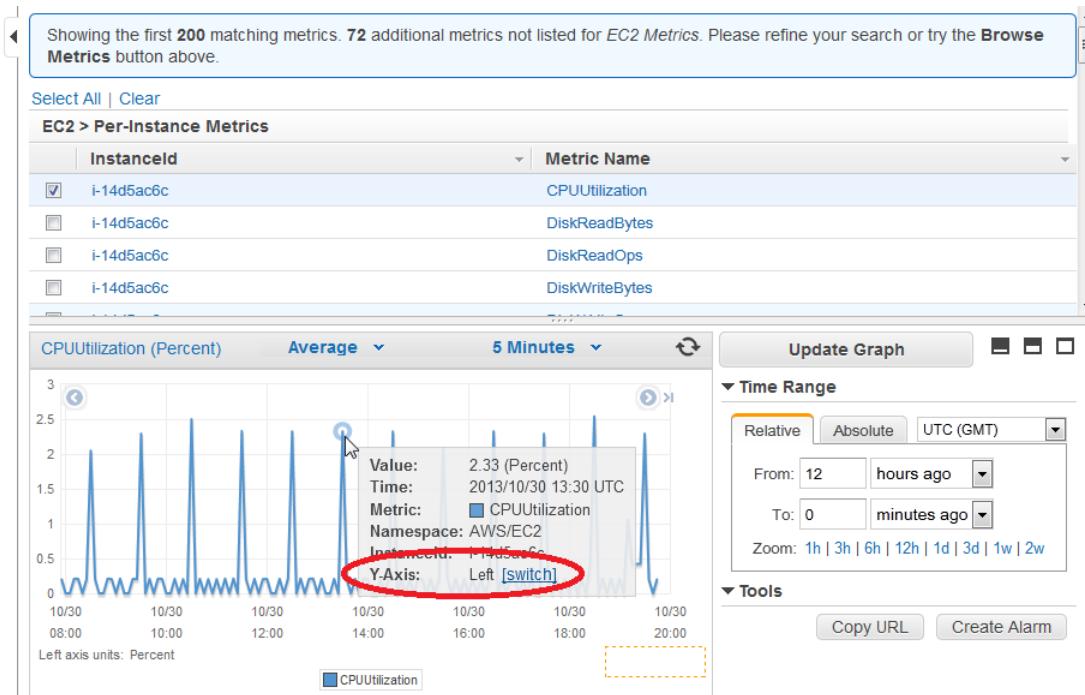
To switch the Y-Axis for a metric.

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, select a category to view available metrics.
5. Select the check box for one or more metrics.
6. In the details pane, hover over a line on the graph or the legend entry for the metric and a hover box appears.

7. Click the **switch** link in the hover box. The metric switches to the opposite axis.



Save a Graph

You can save, or bookmark, a graph to access it later.

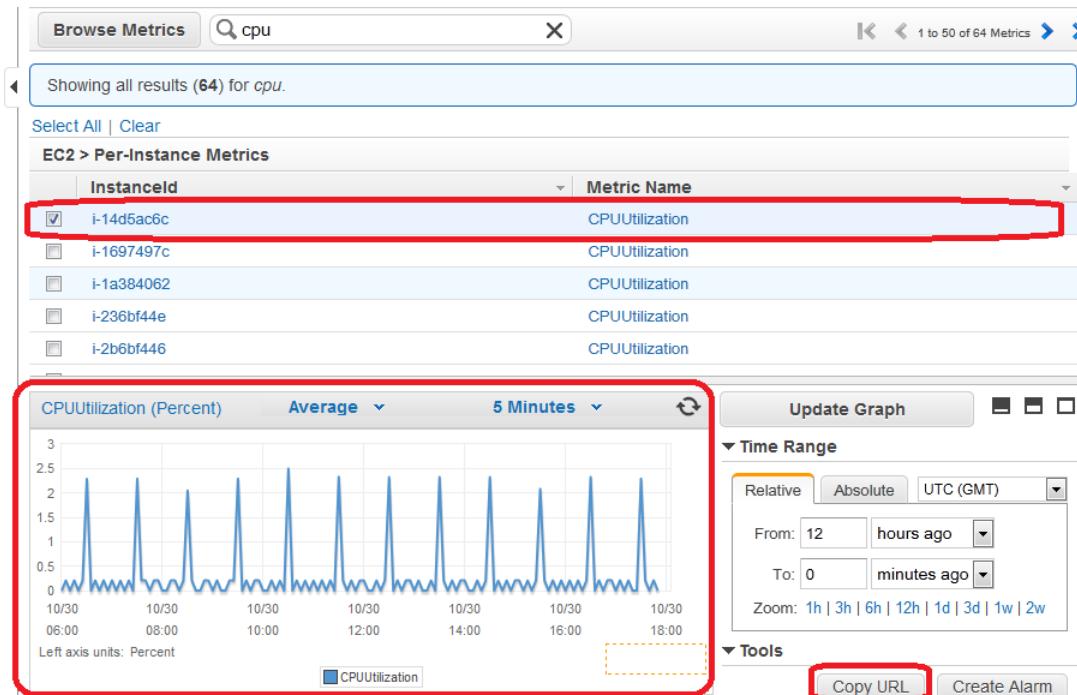
To save a graph

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Metrics**.
4. In the **CloudWatch Metrics by Category** pane, use the **Search Metrics** field and categories to find a metric by metric name, AWS resource, or other metadata.
5. Use the scroll bar and next and previous arrows above the metrics list to page through the full list of metrics

6. Select the metric to view. In the example below, a graph for an Amazon EC2 instance's CPUUtilization metric is displayed in the details pane.



7. In the details pane, under **Tools**, click **Copy URL**.
8. In the **Copy Graph URL** dialog box, select the URL and paste it into your browser.

Using your browser, bookmark the page to access it later.

Publish Custom Metrics

You can publish your own metrics to CloudWatch with the `put-metric-data` command (or its Query API equivalent `PutMetricData`). For more information, see [put-metric-data](#) in the *AWS Command Line Interface Reference*. You can view statistical graphs of your published metrics with the AWS Management Console.

If you call `put-metric-data` with a new metric name, CloudWatch creates a new metric for you. Otherwise, CloudWatch associates your data with the existing metric that you specify.

Note

When you create a new metric using the `put-metric-data` command, it can take up to two minutes before you can retrieve statistics on the new metric using the `get-metric-statistics` command. However, it can take up to fifteen minutes before the new metric appears in the list of metrics retrieved using the `list-metrics` command.

CloudWatch stores data about a metric as a series of data points. Each data point has an associated time stamp. You can publish one or more data points with each call to `put-metric-data`. You can even publish an aggregated set of data points called a *statistics* set.

Topics

- [Publish Single Data Points \(p. 69\)](#)
- [Publish Statistic Sets \(p. 70\)](#)
- [Publish the Value Zero \(p. 70\)](#)

Publish Single Data Points

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

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace "MyService" --value 2 --timestamp 2014-02-14T12:00:00.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace "MyService" --value 4 --timestamp 2014-02-14T12:00:01.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace "MyService" --value 5 --timestamp 2014-02-14T12:00:02.000Z
```

Note

The `put-metric-data` command can only publish one data point per call. If you want to run this example, specify time stamps within the past two weeks.

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

CloudWatch uses one-minute boundaries when aggregating data points. For example, CloudWatch aggregates the data points from the previous example because all three data points fall within the one-minute period that begins at `2014-02-20T12:00:00.000Z` and ends at `2014-02-20T12:05:00.000Z`.

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

```
aws cloudwatch get-metric-statistics --metric-name PageViewCount --namespace "MyService" --statistics "Sum" "Maximum" "Minimum" "Average" "SampleCount" --period 60 --start-time 2014-02-20T12:00:00.000Z --end-time 2014-02-20T12:05:00.000Z --output json
```

CloudWatch returns the following:

```
{
    "Datapoints": [
        {
            "SampleCount": 3.0,
            "Timestamp": "2014-02-20T12:00:00Z",
            "Average": 3.6666666666666665,
            "Maximum": 5.0,
            "Minimum": 2.0,
            "Sum": 11.0,
            "Unit": "None"
        }
    ],
    "Label": "PageViewCount"
}
```

Publish Statistic Sets

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

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace "MyService" --statistic-value Sum=11,Minimum=2,Maximum=5,SampleCount=3 --timestamp 2014-02-14T12:00:00.000Z
```

Publish the Value Zero

When your data is more sporadic and you have periods that have no associated data, you can choose to publish the value zero (0) for that period or no value at all. You might want to publish zero instead of no value if you use periodic calls to `PutMetricData` to monitor the health of your application. For example, you can set an 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.

Creating Amazon CloudWatch Alarms

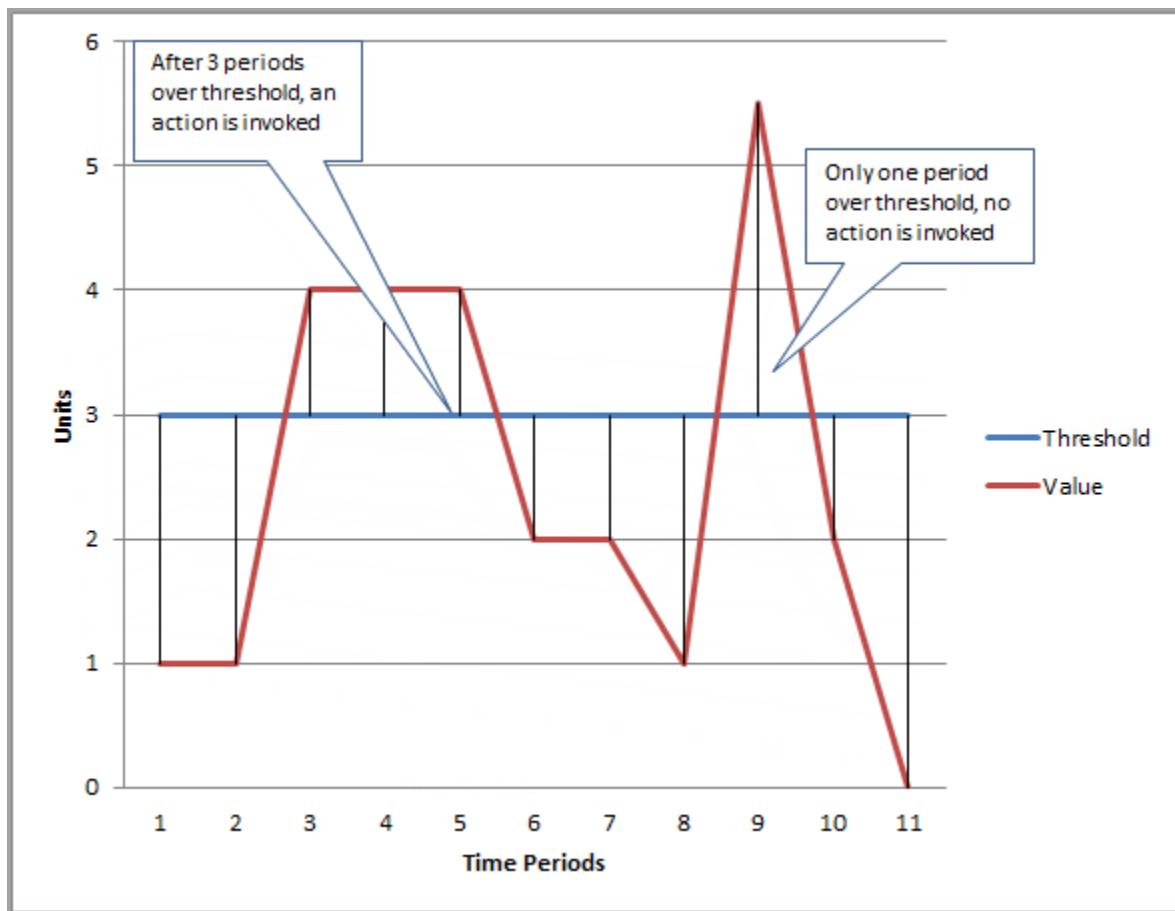
You can create an CloudWatch alarm that sends an Amazon Simple Notification Service message when the alarm changes state. An alarm watches a single metric over a time period you specify, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service topic or Auto Scaling policy. Alarms invoke actions for sustained state changes only. CloudWatch alarms will 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 Auto Scaling policy notifications, the alarm continues to invoke the action for every period that the alarm remains in the new state. For Amazon Simple Notification Service notifications, no additional actions are invoked.

An alarm has three possible states:

- *OK*—The metric is within the defined threshold
- *ALARM*—The metric is outside of the defined threshold
- *INSUFFICIENT_DATA*—The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state

In the following figure, the alarm threshold is set to 3 and the minimum breach is 3 periods. That is, the alarm invokes its action only when the threshold is breached for 3 consecutive periods. In the figure, this happens with the third through fifth time periods, and the alarm's state is set to *ALARM*. At period six, the value dips below the threshold, and the state reverts to *OK*. Later, during the ninth time period, the threshold is breached again, but not for the necessary three consecutive periods. Consequently, the alarm's state remains *OK*.



Note

CloudWatch doesn't test or validate the actions you specify, nor does it detect any Auto Scaling or SNS errors resulting from an attempt to invoke nonexistent actions. Make sure your actions exist.

Common Features of Alarms

- You can create up to 5000 alarms per AWS account. To create or update an alarm, you use the `PutMetricAlarm` API function (`mon-put-metric-alarm` command).
- You can list any or all of the currently configured alarms, and list any alarms in a particular state using the `DescribeAlarms` API (`mon-describe-alarms` command). You can further filter the list by time range.
- You can disable and enable alarms by using the `DisableAlarmActions` and `EnableAlarmActions` APIs (`mon-disable-alarm-actions` and `mon-enable-alarm-actions` commands).
- You can test an alarm by setting it to any state using the `SetAlarmState` API (`mon-set-alarm-state` command). This temporary state change lasts only until the next alarm comparison occurs.
- Finally, you can view an alarm's history using the `DescribeAlarmHistory` API (`mon-describe-alarm-history` command). CloudWatch preserves alarm history for two weeks. Each state transition is marked with a unique time stamp. In rare cases, your history might show more than one notification for a state change. The time stamp enables you to confirm unique state changes.

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 simply be an indication that your resource is inactive, and may not necessarily mean that there is a problem.

Topics

- [Set Up Amazon Simple Notification Service \(p. 73\)](#)
- [Create an Alarm \(p. 78\)](#)
- [Send Email Based on CPU Usage Alarm \(p. 80\)](#)
- [Send Email Based on Load Balancer Alarm \(p. 83\)](#)
- [Send Email Based on Storage Throughput Alarm \(p. 85\)](#)
- [Create Alarms That Stop or Terminate an Instance \(p. 88\)](#)
- [Monitor Your Estimated Charges Using Amazon CloudWatch \(p. 103\)](#)

Set Up Amazon Simple Notification Service

Amazon CloudWatch uses Amazon Simple Notification Service (Amazon SNS) to send email. This section shows you how to create and subscribe to an Amazon Simple Notification Service topic. When you create a CloudWatch alarm, you can add this Amazon SNS topic to send an email notification when the alarm changes state. For more information about Amazon Simple Notification Service, see the [Amazon Simple Notification Service Getting Started Guide](#).

Note

If you create your CloudWatch alarm with the AWS Management Console, you can skip this procedure because you can create an Amazon Simple Notification Service topic in the **Configure Actions** step in the **Create Alarm Wizard**.

Topics

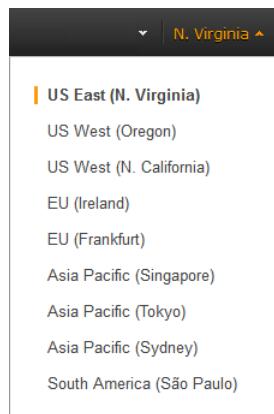
- [AWS Management Console \(p. 73\)](#)
- [Command Line Tools \(p. 77\)](#)

AWS Management Console

To set up an Amazon Simple Notification Service topic with the AWS Management Console first you create a topic, then you subscribe to it. You can then publish a message directly to the topic to ensure that you have properly configured it.

To create an Amazon Simple Notification Service topic

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In **Getting Started**, click **Create New Topic**.

The **Create New Topic** dialog box opens.

The dialog box has a title bar 'Create New Topic' with a 'Cancel' button and a close icon. The main content area contains a descriptive text: 'A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN).'. Below it are two input fields: 'Topic Name *:' and 'Display Name:'. The 'Topic Name *:' field has a placeholder 'Up to 256 alphanumeric characters, hyphens (-) and underscores (_) allowed.' The 'Display Name:' field has a placeholder 'Required for SMS subscriptions (can be up to 10 characters). Optional for other transports.' At the bottom right are 'Cancel' and 'Create Topic' buttons.

4. Enter the topic name *MyTopic* in the **Topic Name** field.
5. Click **Create Topic**.

The new topic appears in the **Topic Details** page.

The 'Topic Details' page for 'MyTopic'. The top navigation bar includes 'Topic Details', 'All Topic Actions', and a 'Publish to Topic' button. The main content area displays the following information:

- Topic ARN:** arn:aws:sns:us-east-1:054794666394:MyTopic
- Topic Owner:** AWS Account ID [REDACTED]
- Region:** us-east-1
- Display Name:** [REDACTED]

At the bottom are buttons for 'Create New Subscription' and 'Delete Subscriptions'.

6. Copy the **Topic ARN** for the next task.

To subscribe to an Amazon Simple Notification Service topic

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **My Subscriptions** in the navigation pane.

The **My Subscriptions** page opens.

4. Click **Create New Subscription**.

The **Subscribe** dialog box opens.

The 'Subscribe' dialog box has the following fields:

- Topic ARN:** A text input field.
- Protocol:** A drop-down menu set to **HTTPS**.
- Endpoint:** A text input field.
- Cancel** and **Subscribe** buttons at the bottom right.

5. In the **Topic ARN** field, paste the topic ARN you created in the previous task, for example:
`arn:aws:sns:us-east-1:054794666394:MyTopic`.
6. Select **Email** in the **Protocol** drop-down list.
7. Enter an email address you can use to receive the notification in the **Endpoint** field, and then click **Subscribe**.

Important

Entourage Users: Entourage strips out the confirmation URL. Please enter an email address you can access in a different email application.

8. Go to your email application and open the message from AWS Notifications, and then click the link to confirm your subscription.

Your web browser displays a confirmation response from Amazon Simple Notification Service.

To publish to an Amazon Simple Notification Service topic

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).

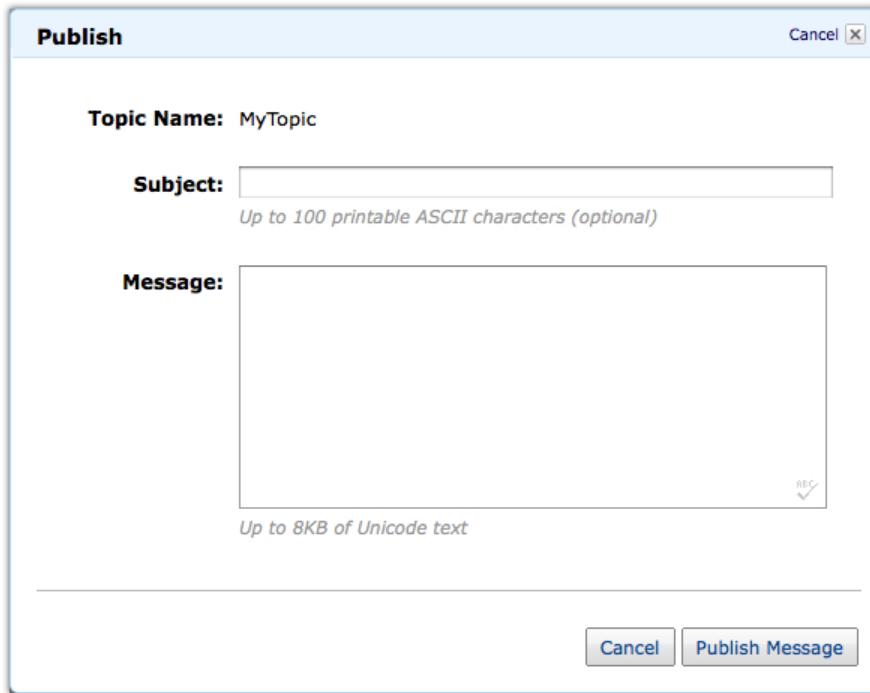


3. In the navigation pane, under **My Topics**, click the topic you want to publish.

The **Topic Details** page opens.

4. Click **Publish to Topic**.

The **Publish** dialog box opens.



5. Enter a subject line for your message in the **Subject** field, and a brief message in the **Message** field.
6. Click **Publish Message**.
A confirmation dialog box opens.
7. Click **Close**.
8. Check your email to confirm that you received the message from the topic.

Command Line Tools

This scenario walks you through how to use the AWS CLI to create an Amazon Simple Notification Service topic, and then publish a message directly to the topic to ensure that you have properly configured it. 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*.

To set up an Amazon Simple Notification Service topic

1. Create the topic using the `create-topic` command. You receive a topic resource name as a return value:

```
Prompt>aws sns create-topic --name MyTopic
```

Amazon Simple Notification Service returns the following Topic ARN:

```
{  
    "TopicArn": "arn:aws:sns:us-east-1:111122223333:MyTopic"  
}
```

2. Subscribe your email address to the topic using the `subscribe` command. You will receive a confirmation email message if the subscription request succeeds.

```
Prompt>aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:MyTopic --protocol email --notification-endpoint <your-email-address>
```

Amazon Simple Notification Service returns the following:

```
{  
    "SubscriptionArn": "pending confirmation"  
}
```

3. Confirm that you intend to receive email from Amazon Simple Notification Service by clicking the confirmation link in the body of the message to complete the subscription process.
4. Check the subscription using the `list-subscriptions-by-topic` command.

```
Prompt>aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-east-1:111122223333:MyTopic
```

Amazon Simple Notification Service returns the following:

```
{  
    "Subscriptions": [  
        {  
            "Owner": "111122223333",  
            "Endpoint": "me@mycompany.com",  
            "Protocol": "email",  
            "TopicArn": "arn:aws:sns:us-east-1:111122223333:MyTopic",  
            "SubscriptionArn": "arn:aws:sns:us-east-1:111122223333:MyTopic:64886986-bf10-48fb-a2f1-dab033aa67a3"  
        }  
    ]  
}
```

5. Publish a message directly to the topic using the `publish` command to ensure that the topic is properly configured.

```
Prompt>aws sns publish --message "Verification" --topic arn:aws:sns:us-east-1:111122223333:MyTopic
```

Amazon Simple Notification Service returns the following:

```
{  
    "MessageId": "42f189a0-3094-5cf6-8fd7-c2dde61a4d7d"  
}
```

6. Check your email to confirm that you received the message from the topic.

Create an Alarm

You can create an alarm from the **Alarms** list in the Amazon CloudWatch console.

To create an alarm

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Alarms**.
4. Click **Create Alarm**, and then in the **CloudWatch Metrics by Category** pane, select a metric category, for example, **EC2 Metrics**.
5. Select a metric, (for example, CPUUtilization), and then click **Next**.
6. Under **Alarm Threshold**, complete the fields, and then under **Actions**, select the type of action you want the alarm to perform when the alarm is triggered.

You can choose specific metrics to trigger the alarm and specify thresholds for those metrics. You can then set your alarm to change state when a metric exceeds a threshold that you have defined. For an example of how to create an alarm that sends email, see [Creating Amazon CloudWatch Alarms](#) (p. 71).

Create Alarm

1. Select Metric 2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name:
Description:

Whenever: CPUUtilization
is: \geq 0
for: 1 consecutive period(s)

Actions

Define what actions are taken when your alarm changes state.

Notification	Delete
Whenever this alarm: State is ALARM Send notification to: Select a notification list <input type="button" value="New list"/>	<input type="button" value="Delete"/>

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line for a duration of 5 minutes

CPUUtilization ≥ 0

Namespace: AWS/EC2
InstanceId: i-0c986c72
Metric Name: CPUUtilization
Period: 5 Minutes
Statistic: Average

7. Click **Create Alarm**.

Send Email Based on CPU Usage Alarm

This scenario walks you through how to use the AWS Management Console or the command line tools to create an Amazon CloudWatch alarm that sends an Amazon Simple Notification Service email message when the alarm changes state from OK to ALARM.

In this scenario, you configure the alarm to change to the ALARM state when the average CPU use of an EC2 instance exceeds 70 percent for two consecutive five-minute periods.

Topics

- [AWS Management Console \(p. 80\)](#)
- [Command Line Tools \(p. 82\)](#)

AWS Management Console

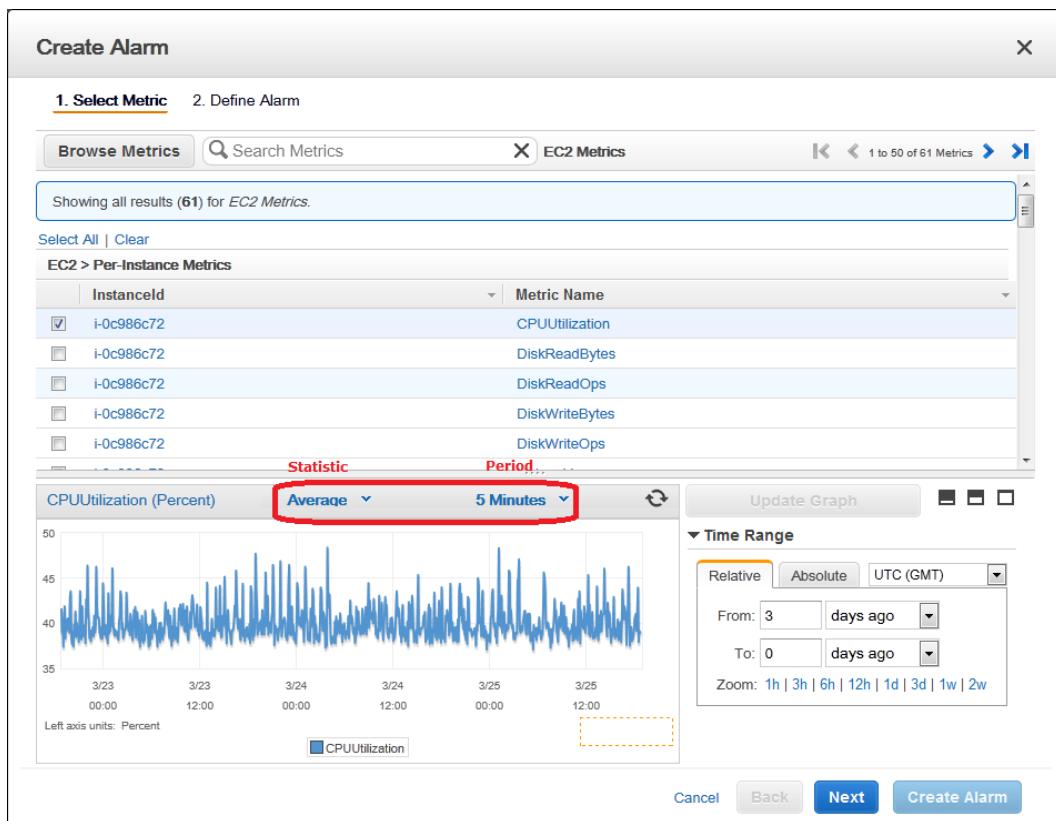
To create an alarm that sends email based on CPU usage

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Alarms**.
4. Click **Create Alarm**, and then in the **CloudWatch Metrics by Category** pane, select a metric category, for example, **EC2 Metrics**.
5. In the list of metrics, select a row that contains **CPUUtilization** for a specific instance ID.

A graph showing average CPUUtilization for a single instance appears in the lower pane.



6. Select **Average** from the **Statistic** drop-down list.
7. Select a period from the **Period** drop-down list, for example: **5 minutes**.
8. Click **Next**, and then under **Alarm Threshold**, in the **Name** field, enter a unique name for the alarm, for example: **myHighCpuAlarm**.
9. In the **Description** field, enter a description of the alarm, for example: **CPU usage exceeds 70 percent**.

10. In the **is** drop-down list, select **>**.
11. In the box next to the **is** drop-down list, enter **70** and in the **for** field, enter **10**.

A graphical representation of the threshold is shown under **Alarm Preview**.
12. Under **Actions**, in the **Whenever this alarm** drop-down list, select **State is ALARM**.
13. In the **Send notification to** drop-down list, select an existing Amazon SNS topic or create a new one.
14. To create a new Amazon SNS topic, select **New list**.

In the **Send notification to** field, enter a name for the new Amazon SNS topic for example: **myHigh-CpuAlarm**, and in the **Email list** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state.
15. In the navigation pane, click **Create Alarm** to complete the alarm creation process.

Command Line Tools

To send an Amazon Simple Notification Service email message when CPU utilization exceeds 70 percent

1. Set up an Amazon Simple Notification Service topic or retrieve the Topic Resource Name of the topic you intend to use. For help on setting up an Amazon Simple Notification Service topic, see [Set Up Amazon Simple Notification Service \(p. 73\)](#).
2. Create an alarm with the `put-metric-alarm` command. Use the values from the following example, but replace the values for `InstanceId` and `alarm-actions` with your own values.

```
Prompt>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:MyTopic --unit Percent
```

The AWS CLI returns to the command prompt if the command succeeds.

3. Test the alarm by forcing an alarm state change with the `set-alarm-state` command.
 - a. Change the alarm state from `INSUFFICIENT_DATA` to `OK`:

```
Prompt>aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason "initializing" --state-value OK
```

The AWS CLI returns to the command prompt if the command succeeds.

- b. Change the alarm state from `OK` to `ALARM`:

```
Prompt>aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason "initializing" --state-value ALARM
```

The AWS CLI returns to the command prompt if the command succeeds.

- c. Check that an email has been received.

Send Email Based on Load Balancer Alarm

This scenario walks you through how to use the Amazon CloudWatch console or the AWS command line interface (CLI) to set up an Amazon Simple Notification Service notification and configure an alarm that monitors load balancer latency exceeding 100 ms.

Topics

- [AWS Management Console \(p. 83\)](#)
- [Command Line Tools \(p. 85\)](#)

AWS Management Console

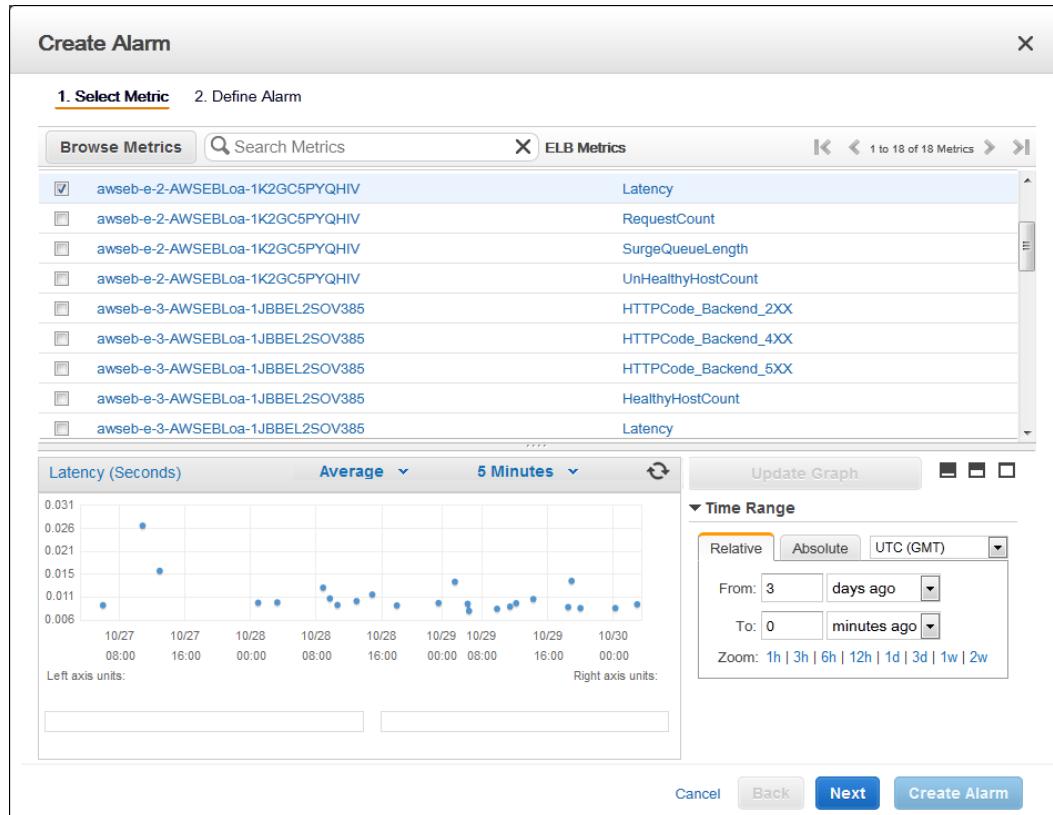
To create a load balancer alarm that sends email

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Alarms**.
4. Click **Create Alarm**, and then in the **CloudWatch Metrics by Category** pane, select a metric category, for example, **ELB Metrics**.
5. In the list of metrics, select a row that contains **Latency** for a specific load balancer.

A graph showing average **Latency** for a single load balancer appears in the lower pane.



6. Select **Average** from the **Statistic** drop-down list.
7. Select **1 Minute** from the **Period** drop-down list.
8. Click **Next**, and then under **Alarm Threshold**, in the **Name** field, enter a unique name for the alarm, for example: **myHighCpuAlarm**.
9. In the **Description** field, enter a description of the alarm, for example: **Alarm when Latency exceeds 100ms**.
10. In the **is** drop-down list, select **>**.
11. In the box next to the **is** drop-down list, enter **0.1** and in the **for** field, enter **3**.

A graphical representation of the threshold is shown under **Alarm Preview**.

12. Under **Actions**, in the **Whenever this alarm** drop-down list, select **State is ALARM**.
13. In the **Send notification to** drop-down list, select an existing Amazon SNS topic or create a new one.
14. To create a new Amazon SNS topic, select **New list**.

In the **Send notification to** field, enter a name for the new Amazon SNS topic for example: **myHighCpuAlarm**, and in the **Email list** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state.

15. In the navigation pane, click **Create Alarm** to complete the alarm creation process.

Command Line Tools

To send an Amazon Simple Notification Service email message when LoadBalancer Latency Exceeds 100 milliseconds

1. Create an Amazon Simple Notification Service topic. See instructions for creating an Amazon SNS topic in [Set Up Amazon Simple Notification Service \(p. 73\)](#)
2. Create the alarm.

```
Prompt>aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm when Latency exceeds 100ms" --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-1:1234567890:my-topic --unit Milliseconds
```

The AWS CLI returns to the command prompt if the command succeeds.

3. Test the alarm.
 - Force an alarm state change to ALARM:

```
Prompt>aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state OK
Prompt>aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state ALARM
```

The AWS CLI returns to the command prompt if the command succeeds.

- Check that an email has been received.

Send Email Based on Storage Throughput Alarm

This scenario walks you through how to use the AWS Management Console or the command line tools to set up an Amazon Simple Notification Service notification and to configure an alarm that sends email when EBS exceeds 100 MB throughput.

Topics

- [AWS Management Console \(p. 85\)](#)
- [Command Line Tools \(p. 87\)](#)

AWS Management Console

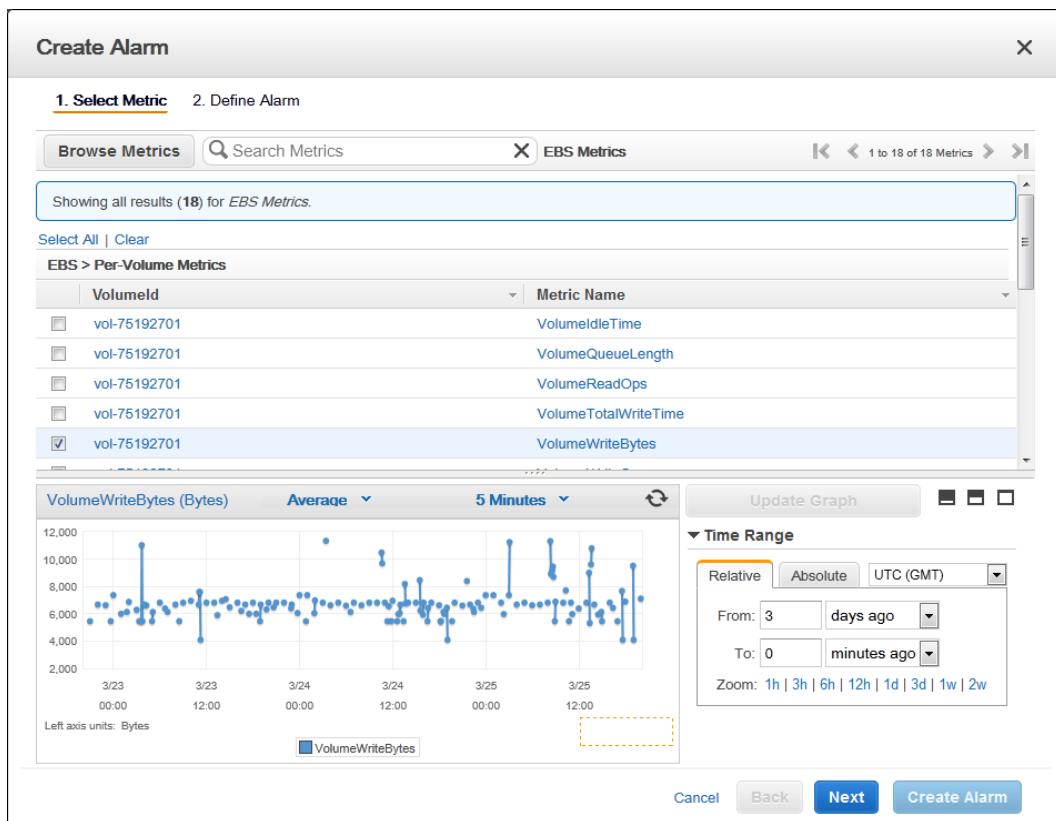
To create a storage throughput alarm that sends email

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Alarms**.
4. Click **Create Alarm**, and then in the **CloudWatch Metrics by Category** pane, select a metric category, for example, **EBS Metrics**.
5. In the list of metrics, select a row that contains **VolumeWriteBytes** for a specific VolumeId.

A graph showing average **VolumeWriteBytes** for a single volume appears in the lower pane.



6. Select **Average** from the **Statistic** drop-down list.
7. Select **5 Minutes** from the **Period** drop-down list.
8. Click **Next**, and then under **Alarm Threshold**, in the **Name** field, enter a unique name for the alarm, for example: **myHighWriteAlarm**.
9. In the **Description** field, enter a description of the alarm, for example: **volumeWriteBytes exceeds 100,000 KiB/s.**

10. In the **is** drop-down list, select **>**.
11. In the box next to the **is** drop-down list, enter **100000** and in the **for** field, enter **15**.
A graphical representation of the threshold is shown under **Alarm Preview**.
12. Under **Actions**, in the **Whenever this alarm** drop-down list, select **State is ALARM**.
13. In the **Send notification to** drop-down list, select an existing Amazon SNS topic or create a new one.
14. To create a new Amazon SNS topic, select **New list**.
In the **Send notification to** field, enter a name for the new Amazon SNS topic for example: **myHigh-CpuAlarm**, and in the **Email list** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state.
15. In the navigation pane, click **Create Alarm** to complete the alarm creation process.

Command Line Tools

To send an Amazon Simple Notification Service email message when EBS exceeds 100 MB throughput

1. Create an Amazon Simple Notification Service topic. See instructions for creating an Amazon SNS topic in [Set Up Amazon Simple Notification Service \(p. 73\)](#).
2. Create the alarm.

```
Prompt>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-1:1234567890:my-alarm-topic --insufficient-data-actions arn:aws:sns:us-east-1:1234567890:my-insufficient-data-topic
```

The AWS CLI returns to the command prompt if the command succeeds.

3. Test the alarm.
 - Force an alarm state change to **ALARM**.

```
Prompt>aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value OK
Prompt>aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value ALARM
Prompt>aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value INSUFFICIENT_DATA
```

- Check that two emails have been received.

Create Alarms That Stop or Terminate an Instance

Using Amazon CloudWatch alarm actions, you can create alarms that automatically stop or terminate your Amazon Elastic Compute Cloud (Amazon EC2) instances when you no longer need them to be running. 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 leave those instances sitting idle (and accruing charges), you can stop or terminate them which can help you to save money. The main difference between using the stop and the terminate alarm actions is that you can easily restart a stopped instance if you need to run it again later, and you can keep the same instance ID and root volume. However, you cannot restart a terminated instance. Instead, you must launch a new instance.

You can add the stop or terminate alarm actions to any alarm that is set on an Amazon EC2 instance metric, including basic and detailed monitoring metrics provided by Amazon CloudWatch (in the AWS/EC2 namespace), as well as any custom metrics that include the “`InstanceId=`” dimension, as long as the `InstanceId` value refers to a valid running Amazon EC2 instance.

Topics

- [Adding Actions to Amazon CloudWatch Alarms \(p. 88\)](#)
- [Amazon CloudWatch Alarm Action Scenarios \(p. 99\)](#)

Adding Actions to Amazon CloudWatch Alarms

You can configure alarm actions using either the Amazon EC2 console or the Amazon CloudWatch console, or you can use the Amazon CloudWatch command line interface (CLI), API, or the AWS SDKs. For information about using the Amazon CloudWatch API with the AWS SDKs, see [Sample Code & Libraries](#).

Using the Amazon EC2 Console to Create an Alarm to Stop an Instance

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 Amazon Simple Notification Service (Amazon 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.

Note

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following Amazon EC2 permissions: `ec2:DescribeInstancesStatus`, `ec2:DescribeInstances`, `ec2:StopInstances`, and `ec2:TerminateInstances` in order for the alarm action to be performed. If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or terminate actions won’t be performed on the Amazon EC2 instance. However, if you are later granted permission to use the associated Amazon EC2 APIs, the alarm actions you created earlier will be performed. For more information about IAM permissions, see [Permissions and Policies in Using IAM](#).

If you are using an IAM role (e.g. Amazon EC2 instance profile), you cannot stop or terminate the instance using alarm actions. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Auto Scaling policies.

If you are using temporary security credentials granted using the AWS Security Token Service (AWS STS), you cannot stop or terminate an Amazon EC2 instance using alarm actions.

To create an alarm to stop an idle instance

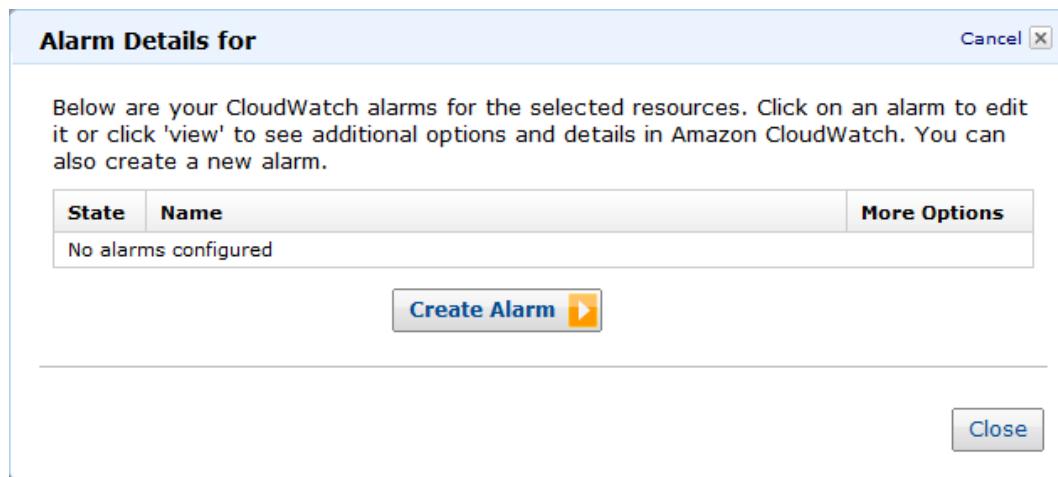
1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. If necessary, change the region. From the navigation bar, select the region where your instance is running. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **INSTANCES**, click **Instances**.
4. In the upper pane, right-click an instance, and then click **Add/Edit Alarms**.

Or, you can also select the instance, and then in the lower pane on the **Monitoring tab**, click **Create Alarm**.

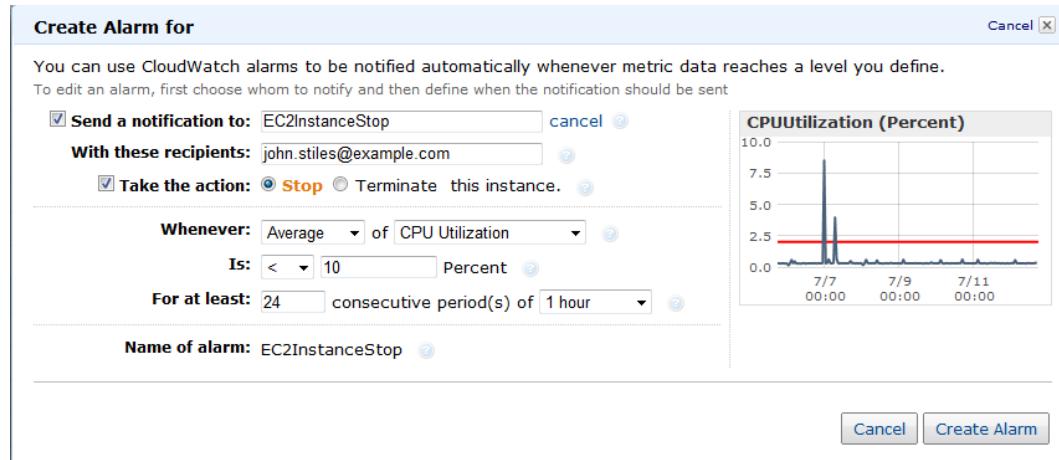
5. In the **Alarm Details for** dialog box, click **Create Alarm**.



6. If you want to receive an email when the alarm is triggered, in the **Create Alarm for** dialog box, in the **Send a notification to** box, select an existing Amazon SNS topic, or click **Create Topic** to create a new one.

If you create a new topic, in the **Send a notification to** box type a name for the topic, and then in the **With these recipients** box, type the email addresses of the recipients (separated by commas).

Later, after you create the alarm, you will receive a subscription confirmation email that you must accept before you will get email for this topic.



7. Select the **Take the action** check box, and then choose the **Stop** radio button.
8. In the **Whenever** boxes, choose the statistic you want to use and then select the metric. In this example, choose **Average** and **CPU Utilization**.
9. In the **Is** boxes, define the metric threshold. In this example, enter **10** percent.
10. In the **For at least** box, choose the sampling period for the alarm. In this example, enter **24** consecutive periods of one hour.
11. To change the name of the alarm, in the **Name this alarm** box, type a new name.

If you don't type a name for the alarm, Amazon CloudWatch will automatically create one for you.

Note

You can adjust the alarm configuration based on your own requirements before creating the alarm, or you can edit them later. This includes the metric, threshold, duration, action, and notification settings. However, after you create an alarm, you cannot edit its name later.

12. Click **Create Alarm**.

Using the Amazon EC2 Console to Create an Alarm that Terminates an Instance

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 on enabling and disabling termination protection for an instance, see [Enabling Termination Protection for an Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following Amazon EC2 permissions: `ec2:DescribeInstancesStatus`, `ec2:DescribeInstances`, `ec2:StopInstances`, and `ec2:TerminateInstances` in order for the alarm action to be performed. If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or terminate actions won't be performed on the Amazon EC2 instance. However, if you are later granted permission to use the associated Amazon EC2 APIs, the alarm actions you created

earlier will be performed. For more information about IAM permissions, see [Permissions and Policies in Using IAM](#).

If you are using an IAM role (e.g. Amazon EC2 instance profile), you cannot stop or terminate the instance using alarm actions. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Auto Scaling policies.

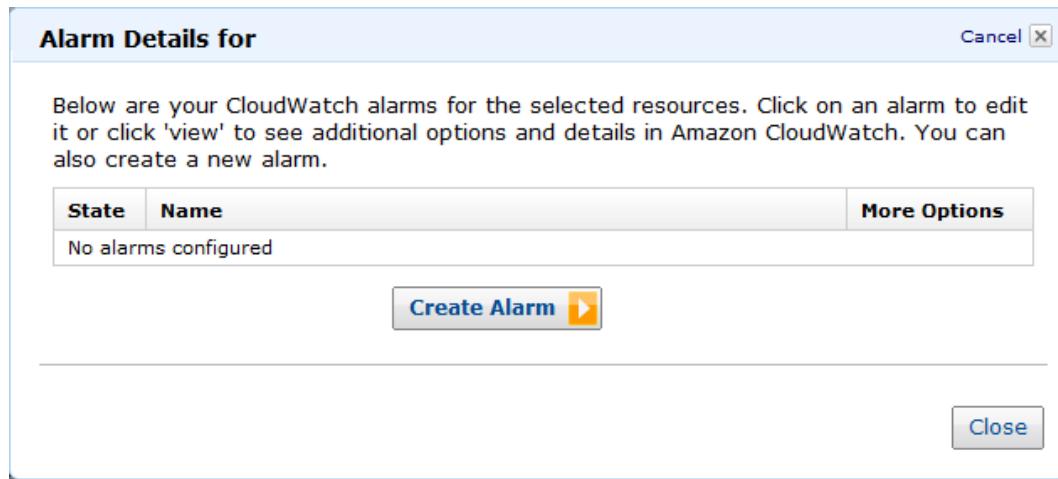
If you are using temporary security credentials granted using the AWS Security Token Service (AWS STS), you cannot stop or terminate an Amazon EC2 instance using alarm actions.

To create an alarm to terminate an idle instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. If necessary, change the region. From the navigation bar, select the region where your instance is running. For more information, see [Regions and Endpoints](#).



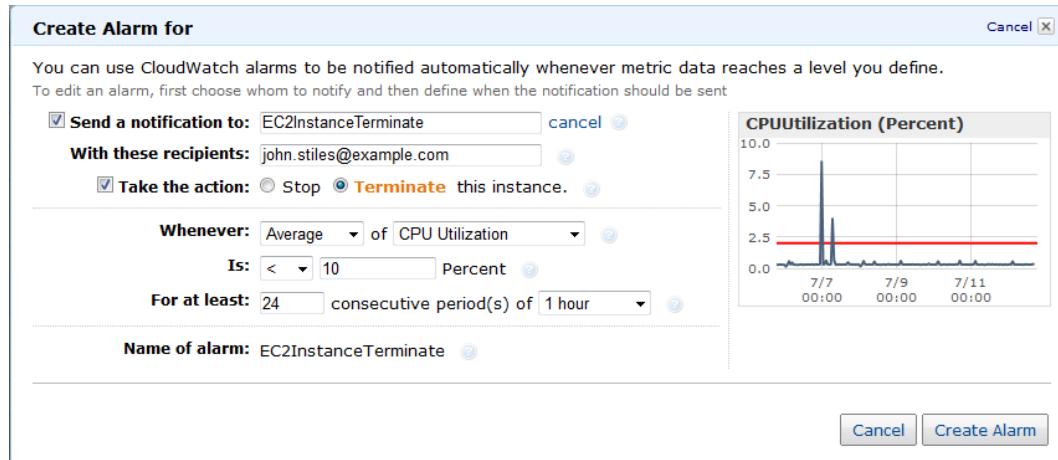
3. In the navigation pane, under **INSTANCES**, click **Instances**.
 4. In the upper pane, right-click an instance, and then click **Add/Edit Alarms**.
- Or, select the instance and then in the lower pane, on the **Monitoring** tab, click **Create Alarm**.
5. In the **Alarm Details for** dialog box, click **Create Alarm**.



6. If you want to receive an email when the alarm is triggered, in the **Create Alarm for** dialog box, in the **Send a notification to** box, select an existing SNS topic, or click **Create Topic** to create a new one.

If you create a new topic, in the **Send a notification to** box type a name for the topic, and then in the **With these recipients** box, type the email addresses of the recipients (separated by commas).

Later, after you create the alarm, you will receive a subscription confirmation email that you must accept before you will get email for this topic.



The screenshot shows the 'Create Alarm for' dialog. Under 'Send a notification to:', 'EC2InstanceTerminate' is selected. 'With these recipients:' is set to 'john.stiles@example.com'. 'Take the action:' is set to 'Terminate'. 'Whenever:' is set to 'Average' of 'CPU Utilization'. 'Is:' is set to '<' 10 Percent. 'For at least:' is set to '24 consecutive period(s)' of '1 hour'. The 'Name of alarm:' is 'EC2InstanceTerminate'. To the right, there's a preview chart titled 'CPUUtilization (Percent)' showing a single data series with a red horizontal line at 10% and a sharp peak above it around July 7th.

7. Select the **Take the action** check box, and then choose the **Terminate** radio button.
8. In the **Whenever** boxes, choose the statistic you want to use and then select the metric. In this example, choose **Average** and **CPU Utilization**.
9. In the **Is** boxes, define the metric threshold. In this example, enter **10** percent.
10. In the **For at least** box, choose the sampling period for the alarm. In this example, enter **24** consecutive periods of one hour.
11. To change the name of the alarm, in the **Name this alarm** box, type a new name.

If you don't type a name for the alarm, Amazon CloudWatch will automatically create one for you.

Note

You can adjust the alarm configuration based on your own requirements before creating the alarm, or you can edit them later. This includes the metric, threshold, duration, action, and notification settings. However, after you create an alarm, you cannot edit its name later.

12. Click **Create Alarm**.

Using the Amazon CloudWatch Console to Create an Alarm that Stops an Instance

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 Amazon Simple Notification Service (Amazon SNS) notification, so that you will receive an email when the alarm is triggered.

Amazon CloudWatch alarm actions can stop an EBS-backed Amazon EC2 instances but they cannot stop instance store-backed Amazon EC2 instances. However, Amazon CloudWatch alarm actions can terminate either type of Amazon EC2 instance.

Note

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following Amazon EC2 permissions: `ec2:DescribeInstancesStatus`, `ec2:DescribeInstances`, `ec2:StopInstances`, and `ec2:TerminateInstances` in order for the alarm action to be performed. If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or ter-

minate actions won't be performed on the Amazon EC2 instance. However, if you are later granted permission to use the associated Amazon EC2 APIs, the alarm actions you created earlier will be performed. For more information about IAM permissions, see [Permissions and Policies in Using IAM](#).

If you are using an IAM role (e.g. Amazon EC2 instance profile), you cannot stop or terminate the instance using alarm actions. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Auto Scaling policies.

If you are using temporary security credentials granted using the AWS Security Token Service (AWS STS), you cannot stop or terminate an Amazon EC2 instance using alarm actions.

To create an alarm to stop an idle instance

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region where your instance is running. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Alarms**.
4. Click **Create Alarm**, and then in **CloudWatch Metrics by Category**, under **EC2 Metrics**, select **Per-Instance Metrics**.
5. In the list of metrics, select the instance and metric you want to create an alarm for. You can also type an instance ID in the search box to go the instance that you want.
6. Select **Average** from the **Statistic** drop-down list.
7. Select a period from the **Period** drop-down list, for example: **1 Day**.
8. Click **Next**, and then under **Alarm Threshold**, in the **Name** field, enter a unique name for the alarm, for example: **Stop EC2 instance**.
9. In the **Description** field, enter a description of the alarm, for example: **Stop EC2 instance when CPU is idle for too long**.
10. In the **is** drop-down list, select **<**.
11. In the box next to the **is** drop-down list, enter **10** and in the **for** field, enter **1440**.

A graphical representation of the threshold is shown under **Alarm Preview**.

12. Under **Actions**, click **EC2 Action**.
13. In the **Whenever this alarm** drop-down list, select **State is ALARM**.
14. In the **Take this action** drop-down list, select **Stop this instance**.
15. Click **Notification**, and then in the **Send notification to** drop-down list, select an existing Amazon SNS topic or create a new one.
16. To create a new Amazon SNS topic, select **New list**.

In the **Send notification to** field, enter a name for the new Amazon SNS topic for example: **Stop_EC2_Instance**, and in the **Email list** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state.

Important

If you are creating a new topic or adding email addresses to an existing topic, each email address that you add will be sent a topic subscription confirmation email. You must confirm the subscription by clicking the included link before notifications will be sent to a new email address.

17. In the navigation pane, click **Create Alarm** to complete the alarm creation process.

Using the Amazon CloudWatch Console to Create an Alarm to Terminate an Idle Instance

You can create an alarm that terminates an Amazon EC2 instance automatically when a certain threshold has been met, as long as termination protection is disabled on 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 on disabling termination protection on an instance, see [Enabling Termination Protection for an Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following Amazon EC2 permissions: `ec2:DescribeInstancesStatus`, `ec2:DescribeInstances`, `ec2:StopInstances`, and `ec2:TerminateInstances` in order for the alarm action to be performed. If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or terminate actions won't be performed on the Amazon EC2 instance. However, if you are later granted permission to use the associated Amazon EC2 APIs, the alarm actions you created earlier will be performed. For more information about IAM permissions, see [Permissions and Policies in Using IAM](#).

If you are using an IAM role (e.g. Amazon EC2 instance profile), you cannot stop or terminate the instance using alarm actions. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Auto Scaling policies.

If you are using temporary security credentials granted using the AWS Security Token Service (AWS STS), you cannot stop or terminate an Amazon EC2 instance using alarm actions.

To create an alarm to terminate an idle instance

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region where your instance is running. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Alarms**.
4. Click **Create Alarm**, and then in **CloudWatch Metrics by Category**, under **EC2 Metrics**, select **Per-Instance Metrics**.

5. In the list of metrics, select the instance and metric you want to create an alarm for. You can also type an instance ID in the search box to go the instance that you want.
6. Select **Average** from the **Statistic** drop-down list.
7. Select a period from the **Period** drop-down list, for example: **1 Day**.
8. Click **Next**, and then under **Alarm Threshold**, in the **Name** field, enter a unique name for the alarm, for example: **Terminate EC2 instance**.
9. In the **Description** field, enter a description of the alarm, for example: **Terminate EC2 instance when CPU is idle for too long**.
10. In the **is** drop-down list, select **<**.
11. In the box next to the **is** drop-down list, enter **10** and in the **for** field, enter **1440**.

A graphical representation of the threshold is shown under **Alarm Preview**.

12. Under **Actions**, click **EC2 Action**.
13. In the **Whenever this alarm** drop-down list, select **State is ALARM**.
14. In the **Take this action** drop-down list, select **Terminate this instance**.
15. Click **Notification**, and then in the **Send notification to** drop-down list, select an existing Amazon SNS topic or create a new one.
16. To create a new Amazon SNS topic, select **New list**.

In the **Send notification to** field, enter a name for the new Amazon SNS topic for example: **Terminate_EC2_Instance**, and in the **Email list** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state.

Important

If you are creating a new topic or adding email addresses to an existing topic, each email address that you add will be sent a topic subscription confirmation email. You must confirm the subscription by clicking the included link before notifications will be sent to a new email address.

17. In the navigation pane, click **Create Alarm** to complete the alarm creation process.

Using the Amazon CloudWatch Console to View 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 Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region where your instance is running. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, click **Alarms**.
4. In the upper pane, select the alarm with the history that you want to view.
5. In the lower pane, the **Details** tab shows the most recent state transition along with the time and metric values.
6. Click the **History** tab to view the most recent history entries.

Using the CLI or the API to Create an Alarm to Stop or Terminate an Instance

If you are using either the AWS CLI or the CloudWatch API, or if you are using the AWS SDKs with the API, you can create an Amazon CloudWatch alarm using an Amazon EC2 instance metric, and then add an action using the action's dedicated Amazon Resource Name (ARN). You can add the action to any alarm state, and you can specify the region for each action. The region must match the region to which you send the put-metric-alarm request.

Action	ARN (with region)
Stop	arn:aws:automate:us-east-1:ec2:stop
Terminate	arn:aws:automate:us-east-1:ec2:terminate

For information about using the Amazon CloudWatch API with the AWS SDKs, see [Sample Code & Libraries](#).

Note

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following Amazon EC2 permissions: `ec2:DescribeInstanceStatus`, `ec2:DescribeInstances`, `ec2:StopInstances`, and `ec2:TerminateInstances` in order for the alarm action to be performed. If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or terminate actions won't be performed on the Amazon EC2 instance. However, if you are later granted permission to use the associated Amazon EC2 APIs, the alarm actions you created earlier will be performed. For more information about IAM permissions, see [Permissions and Policies in Using IAM](#).

If you are using an IAM role (e.g. Amazon EC2 instance profile), you cannot stop or terminate the instance using alarm actions. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Auto Scaling policies.

If you are using temporary security credentials granted using the AWS Security Token Service (AWS STS), you cannot stop or terminate an Amazon EC2 instance using alarm actions.

To create an alarm to stop an instance using the CLI

You can use the `arn:aws:automate:us-east-1:ec2:stop` ARN to stop an Amazon EC2 instance. The following example shows how to stop an instance if the average CPUUtilization is less than 10 percent over a 24 hour period.

- At a command prompt, type:

```
% aws cloudwatch put-metric-alarm --alarm-name my-Alarm --alarm-description "Stop the instance when it is idle for a day" --namespace "AWS/EC2" --dimensions Name=InstanceId,Value=i-abc123 --statistic Average --metric-name CPUUtilization --comparison-operator LessThanThreshold --threshold 10 --period 86400 --evaluation-periods 4 --alarm-actions arn:aws:automate:us-east-1:ec2:stop
```

To create an alarm to terminate an instance using the CLI

- At a command prompt, type:

```
% aws cloudwatch put-metric-alarm --alarm-name my-Alarm --alarm-description "Terminate the instance when it is idle for a day" --namespace "AWS/EC2" --dimensions Name=InstanceId,Value=i-abc123" --statistic Average --metric-name CPUUtilization --comparison-operator LessThanThreshold --threshold 1 --period 86400 --evaluation-periods 4 --alarm-actions arn:aws:automate:us-east-1:ec2:terminate
```

To create an alarm and to stop an instance using the API

The following example request shows how to create an alarm that stops an Amazon EC2 instance.

- Construct the following request:

```
http://monitoring.amazonaws.com/  
?SignatureVersion=2  
&Action=PutMetricAlarm  
&Version=2009-05-15  
&Namespace=AWS/EC2  
&MetricName=CPUUtilization  
&Dimension.member.1.Name=instance-id  
&Dimension.member.1.Value=i-abc123  
&Period=86400  
&Statistic=Average  
&AlarmName=Stop-EC2-Instance
```

```
&ComparisonOperator=LessThanThreshold  
&Threshold=10  
&EvaluationPeriods=4  
&StartTime=2009-01-16T00:00:00  
&EndTime=2009-01-16T00:02:00  
&Timestamp=2009-01-08-18  
&AWSAccessKeyId=XXX YOUR ACCESS KEY XXX  
&Signature=%XXX YOUR SIGNATURE XXX%3D  
&AlarmActions.member.1=arn:aws:automate:us-east-1:ec2:stop
```

To create an alarm and to terminate an instance using the API

The following example request shows how to create an alarm that terminates an Amazon EC2 instance.

- Construct the following request:

```
http://monitoring.amazonaws.com/  
?SignatureVersion=2  
&Action=PutMetricAlarm  
&Version=2009-05-15  
&Namespace=AWS/EC2  
&MetricName=CPUUtilization  
&Dimension.member.1.Name=instance-id  
&Dimension.member.1.Value=i-abc123  
&Period=86400  
&Statistic=Average  
&AlarmName=Terminate-EC2-Instance  
&ComparisonOperator=LessThanThreshold  
&Threshold=10  
&EvaluationPeriods=4  
&StartTime=2009-01-16T00:00:00
```

```
&EndTime=2009-01-16T00:02:00
&Timestamp=2009-01-08-18
&AWSAccessKeyId=XXX YOUR ACCESS KEY XXX
&Signature=%XXX YOUR SIGNATURE XXX%3D
&AlarmActions.member.1=arn:aws:automate:us-east-1:ec2:terminate
```

Amazon CloudWatch Alarm Action Scenarios

You can use the Amazon Elastic Compute Cloud (Amazon EC2) console to create alarm actions that stop or terminate an Amazon EC2 instance when certain conditions are met. In the following screen capture of the console page where you set the alarm actions, we've numbered the settings. We've also numbered the settings in the scenarios that follow, to help you create the appropriate actions.

Scenario 1: Stop Idle Development and Test Instances

Create an alarm that stops an instance used for software development or testing when it has been idle for at least an hour.

Seq	Value
①	Stop
②	Maximum
③	CPUUtilization
④	<=
⑤	10%
⑥	60 minutes

Sig	Value
7	1

Scenario 2: Stop Idle Instances

Create an alarm that stops an instance and sends an email when the instance has been idle for 24 hours.

Sig	Value
1	Stop and email
2	Average
3	CPUUtilization
4	<=
5	5%
6	60 minutes
7	24

Scenario 3: Stop Web Servers with Unusually High Traffic

Create an alarm that sends email when an instance exceeds 10 GB of outbound network traffic per day.

Sig	Value
1	Email
2	Sum
3	NetworkOut
4	>
5	10 GB
6	1 day
7	1

Scenario 4: Stop Web Servers with Unusually High Traffic

Create an alarm that stops an instance and send a text message (SMS) if outbound traffic exceeds 1 GB per hour.

Sig	Value
①	Stop and send SMS
②	Sum
③	NetworkOut
④	>
⑤	1 GB
⑥	1 hour
⑦	1

Scenario 5: Stop an Instance Experiencing a Memory Leak

Create an alarm that stops an instance when memory utilization reaches or exceeds 90%, so that application logs can be retrieved for troubleshooting.

Note

The MemoryUtilization metric is a custom metric. In order to use the MemoryUtilization metric, you must install the [Monitoring Scripts for Amazon EC2 Instances \(p. 153\)](#).

Sig	Value
①	Stop
②	Maximum
③	MemoryUtilization
④	>=
⑤	90%
⑥	1 minute
⑦	1

Scenario 6: Stop an Impaired Instance

Create an alarm that stops an instance that fails three consecutive status checks (performed at 5-minute intervals).

Sig	Value
①	Stop

Setting	Value
②	Average
③	StatusCheckFailed_System
④	>=
⑤	1
⑥	15 minutes
⑦	1

Scenario 7: Terminate Instances When Batch Processing Jobs Are Complete

Create an alarm that terminates an instance that runs batch jobs when it is no longer sending results data.

Setting	Value
①	Terminate
②	Maximum
③	NetworkOut
④	<=
⑤	100,000 bytes
⑥	5 minutes
⑦	1

The previous scenarios can also be performed using the Amazon CloudWatch console. We've numbered the settings on the console to match the numbered settings in the Amazon EC2 console and the scenarios that we covered earlier, so you can make a comparison and create an alarm with the appropriate actions.

Create Alarm

1. Select Metric 2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name:
 Description:

Whenever: CPUUtilization
 is: 4 5
 for: 7 consecutive period(s)

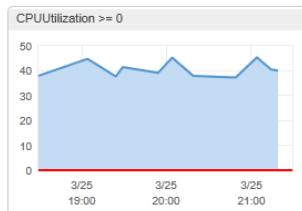
Actions

Define what actions are taken when your alarm changes state.

Notification	Delete
Whenever this alarm: State is ALARM	Delete
① Send notification to: <input type="text"/> <input type="button" value="New list"/>	
<input type="button" value="+ Notification"/> <input type="button" value="+ AutoScaling Action"/> <input type="button" value="+ EC2 Action"/>	

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line for a duration of 5 minutes



CPUUtilization >= 0

Namespaces: AWS/EC2
 InstanceID: i-0c986c72
 ③ Metric Name: CPUUtilization
 ⑥ Period: 5 Minutes
 ② Statistic: Average

Monitor Your Estimated Charges Using Amazon CloudWatch

You can monitor your estimated Amazon Web Services (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 Amazon CloudWatch as metric data that is stored for 14 days. Billing metric data is stored in the US East (N. Virginia) region and represent 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 Amazon CloudWatch and are sent using Amazon Simple Notification Service (Amazon SNS) notification.

The metrics are provided free of charge, and you get 10 Amazon CloudWatch alarms and 1,000 Amazon SNS email notifications per customer per month for free. Any additional alarms or email notifications are priced at standard AWS rates. For more information, see [Amazon CloudWatch Pricing](#), and [Amazon SNS Pricing](#).

Topics

- [Enabling the Monitoring of Your Estimated Charges \(p. 104\)](#)
- [Creating a Billing Alarm \(p. 104\)](#)
- [Editing a Billing Alarm \(p. 110\)](#)
- [Checking Alarm Status \(p. 110\)](#)
- [Deleting a Billing Alarm \(p. 111\)](#)

Enabling the Monitoring of Your Estimated Charges

Before you can create an alarm on your estimated charges, you must enable monitoring, which creates metric data that you can use to create a billing alarm. It takes about 15 minutes before you can view billing data and create alarms using the Amazon CloudWatch console or command line interface (CLI). You must be signed in as the account owner (the "root user") to enable billing alerts for your AWS account. After you enable billing metrics you cannot disable the collection of data, but you can delete any alarms you have created.

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 spaces provided, enter your user name and password, and then click **Sign in using our secure server**.
3. In the navigation pane, click **Preferences**, and then select the **Receive Billing Alerts** check box.

Dashboard

Bills

Payment Methods

Payment History

Consolidated Billing

Account Settings

Reports

Preferences

Credits

Preferences

Receive PDF Invoice By Email

Turn on this feature to receive a PDF version of your invoice by email. Invoices are generally available within the first three days of the month.

Receive Billing Alerts

Turn on this feature to monitor your AWS usage charges and recurring fees automatically, making it easier to track and manage your spending on AWS. You can set up billing alerts to receive email notifications when your charges reach a specified threshold. Once enabled, this preference cannot be disabled.

Receive Billing Reports

Turn on this feature to receive ongoing reports of your AWS charges once or more daily. AWS delivers these reports to the Amazon S3 bucket that you specify where indicated below. For consolidated billing customers, AWS generates reports only for paying accounts. Linked accounts cannot sign up for billing reports.

Save to S3 Bucket:

Note: You must apply appropriate permissions to your S3 bucket [sample policy](#)

Creating a Billing Alarm

You can use the Amazon CloudWatch console or the AWS command line interface (CLI) to create a billing alarm. You can apply more than one action to an alarm, or you can set the alarm to be triggered so that you receive an email when charges for a specified service exceed a certain amount.

To create a billing alarm using the Amazon CloudWatch console

In this example, you can graph your charges for Amazon EC2 over the last 14 days and then set an alarm that sends email as soon as your charges exceed \$200.

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, click **Alarms**, and then in the **Alarms** pane, click **Create Alarm**.

3. In the **CloudWatch Metrics by Category** pane, under **Billing Metrics**, click **By Service**.

The screenshot shows the 'Create Alarm' wizard with the title 'Create Alarm' at the top. Below it, there are two tabs: '1. Select Metric' (which is selected) and '2. Define Alarm'. A search bar labeled 'Search Metrics' is present. The main content area is titled 'CloudWatch Metrics by Category' and displays a summary of metrics:

CloudWatch Metrics by Category		
Your CloudWatch metric summary has loaded. Total metrics: 166		
Billing Metrics: 9	EBS Metrics: 36	EC2 Metrics: 61
Total Estimated Charge: 1	Per-Volume Metrics: 36	Per-Instance Metrics: 40
By Service: 8		By Image (AMI) Id: 7
		Aggregated by Instance Type: 7
		Across All Instances: 7
RDS Metrics: 52	SNS Metrics: 8	
Per-Database Metrics: 13	Topic Metrics: 8	
By Database Class: 13		
By Database Engine: 13		
Across All Databases: 13		

At the bottom right, there are buttons for 'Update Graph', 'Cancel', 'Back', 'Next', and 'Create Alarm'.

4. Under **Billing > By Service**, select a service (e.g., Amazon EC2) to view its billing data in a graph in the lower pane.

Create Alarm

1. Select Metric 2. Define Alarm

Browse Metrics
Search Metrics
X Billing > By Service
1 to 8 of 8 Metrics

Showing all results (8) for *Billing > By Service*.

Select All | Clear

ServiceName	Currency	Metric Name
AWSDataTransfer	USD	EstimatedCharges
AWSDirectConnect	USD	EstimatedCharges
AWSSupportDeveloper	USD	EstimatedCharges
<input checked="" type="checkbox"/> AmazonEC2	USD	EstimatedCharges
AmazonRDS	USD	EstimatedCharges

EstimatedCharges (None) Maximum 6 Hours

Left axis units: None

EstimatedCharges

Update Graph

Time Range

Relative Absolute UTC (GMT)

From: 12 hours ago

To: 0 minutes ago

Zoom: 1h | 3h | 6h | 12h | 1d | 3d | 1w | 2w

[Cancel](#) [Back](#) [Next](#) **Create Alarm**

Note

There are several different ways you can view billing data for your account - **Total Estimated Charges**, **By Service**, or **By Linked 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.

5. Click **Next**, and then under **Alarm Threshold**, in the **Name** box, enter a unique, friendly name for the alarm (for example, My Estimated Charges).

Create Alarm

1. Select Metric 2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name:	<input type="text"/>
Description:	<input type="text"/>
Whenever charges for: EstimatedCharges	
is:	>= <input type="text"/> USD \$ <input type="text"/>

Actions

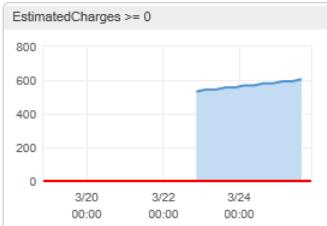
Define what actions are taken when your alarm changes state.

Notification	Delete
Whenever this alarm:	<input type="button" value="State is ALARM"/>
Send notification to:	<input type="button" value="Select a notification list"/> New list

[+ Notification](#) [+ AutoScaling Action](#) [+ EC2 Action](#)

Alarm Preview

This alarm will trigger when the **blue line** goes up to or above the **red line**



EstimatedCharges >= 0

3/20	3/22	3/24
0:00	0:00	0:00

Namespace: AWS/Billing
ServiceName: AmazonEC2
Currency: USD
Metric Name: EstimatedCharges

[Cancel](#) [Back](#) [Next](#) [Create Alarm](#)

6. In the **Description** box, enter a description for the alarm (for example, Estimated Monthly Charges).
7. Under **Whenever charges for**, in the **is** drop-down list, select **>=** (greater than or equal to), and then in the **USD** box, set the monetary amount (for example, 200) that must be exceeded to trigger the alarm and send an email.

Note

Under **Alarm Preview**, in the **Estimated Monthly Charges** thumbnail graph, you can see an estimate of your charges that you can use to set an appropriate threshold for the alarm.

8. Under **Actions**, click **Notification**, and then in the **Whenever this alarm** drop-down menu, click **State is ALARM**.
9. In the **Send notification to** box, select an existing Amazon SNS topic.

To create a new Amazon SNS topic, click **New list**, and then in the **Send notification to** box, enter a name for the new Amazon SNS topic (for example, CFO), and in the **Email list** box, enter the email address (for example, john.stiles@example.com) where email notifications should be sent.

Note

If you create a new Amazon SNS topic, the email account associated with the topic will receive a subscription confirmation email. You must confirm the subscription in order to receive future email notifications when the alarm is triggered.

1. Select Metric

2. Define Alarm

[Back](#) [Next](#)

[Cancel](#)

Please set the alarm threshold, actions and click **Create Alarm** below.

Create Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name:	<input type="text" value="My Estimated Charges"/>
Description:	<input type="text" value="Estimated Monthly Charges"/>
Whenever charges for: EstimatedCharges	
is:	>= <input type="text" value="USD \$ 200"/>
for:	<input type="text" value="1"/> consecutive period(s)

Actions

Define what actions are taken when your alarm changes state.

Notification	Delete
Whenever this alarm:	<input type="text" value="State is ALARM"/>
Send notification to:	<input type="text" value="CFO"/>
Email list:	<input type="text" value="john.stiles@example.com"/> ...

[+ Notification](#) [+ AutoScaling Action](#) [+ EC2 Action](#)

Alarm Preview

This alarm will trigger when the **blue line** goes up to or above the **red line** for a duration of **6 hours**

EstimatedCharges >= 200

11/8 00:00	11/10 00:00	11/12 00:00
0	100	200
200	300	400
300	400	450

Namespace: AWS/Billing
ServiceName: AmazonEC2
Currency: USD
Metric Name: EstimatedCharges

Period: 6 Hours
Statistic: Maximum

10. Click **Create Alarm**.

Important

If you added an email address to the list of recipients or created a new topic, Amazon SNS sends a subscription confirmation email to each new address shortly after you create an alarm. Remember to click the link contained in that message, which confirms your subscription. Alert notifications are only sent to confirmed addresses.

11. To view your billing alarm in the CloudWatch console, in the navigation pane, under **Alarms**, click **Billing**.

To create a billing alarm using the CLI

In addition to using the Amazon CloudWatch console, you can also use the AWS command line interface (CLI) to create a billing alarm for any service in AWS that you're using. You can apply more than one action to an alarm, or you can set the alarm to be triggered so you receive an email when charges for a specified service fall below a certain amount.

The example in the following procedure creates an alarm that will send an email message when your estimated month-to-date charges for Amazon EC2 exceed \$50.

1. At a command prompt, type `aws cloudwatch list-metrics` to view the list of all available Amazon CloudWatch metrics for the services in AWS that you're using.
2. In the list of metrics, review the billing metrics that have the AWS/Billing namespace. These are the billing metrics that you can use to create a billing alarm.
3. At the command prompt, enter the following command:

```
aws cloudwatch put-metric-alarm --alarm-name ec2billing --comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --metric-name EstimatedCharges --namespace AWS/Billing --dimensions Name=Currency,Value=USD" --
```

```
period 21600 --statistic Maximum --threshold 50 --actions-enabled --alarm-actions arn:aws:sns:us-east-1:111111111111:NotifyMe
```

Where:

The **--alarm-name** is the name that you want to give the alarm.

The **--comparison-operator** is one of the following values: GreaterThanOrEqualToThreshold, GreaterThanThreshold, LessThanThreshold, or LessThanOrEqualToThreshold.

The **--evaluation-periods** are the number of periods over which data is compared to the specified threshold (e.g., One period equals 6 hours).

The **--metric-name** is one of the available billing metrics (e.g., EstimatedCharges).

The **--namespace** is the metric's namespace (e.g., AWS/Billing).

The **--dimensions** are associated with the metric (e.g., Currency=USD).

The **--period** is the time frame (in seconds) in which Amazon CloudWatch metrics are collected. In this example, you would enter 21600, which is 60 seconds multiplied by 60 minutes multiplied by 6 hours.

The **--statistic** is one of the following values: SampleCount, Average, Sum, Minimum, or Maximum.

The **--threshold** is the dollar amount you want to use e.g., 50.

The **--actions-enabled** indicates that the alarm should perform an action. Use `--no-actions-enabled` to disregard the `--alarm-actions`.

The **--alarm-actions** is the list of actions to perform when this alarm is triggered. Each action is specified as an Amazon Resource Name (ARN). In this example, we want the alarm to send us an email using Amazon SNS.

Note

You can find the ARN for the Amazon SNS topic that the alarm will use in the Amazon SNS console:

- a. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/>.
- b. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



- c. On the navigation pane, under **My Topics**, select the topic you want the alarm to send mail to.
- d. The ARN is located in the **Topic ARN** field on the **Topic Details** pane.

Editing a Billing Alarm

You can edit an existing billing alarm and make changes to it using the Amazon CloudWatch console or the AWS command line interface (CLI).

To edit a billing alarm using the Amazon CloudWatch console

1. Sign in to the AWS Management Console and 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 the US East (N. Virginia) region and represent worldwide charges. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **Alarms**, click **Billing**.
4. In the list of alarms, select the check box next to the alarm you want to change, and then click **Modify**.
5. Under **Alarm Threshold**, in the **USD** box, set the monetary amount (for example, 400) that must be exceeded to trigger the alarm and send an email, and then in the navigation pane, click **Save Changes**.

To edit a billing alarm using the CLI

1. At a command prompt, type `aws cloudwatch describe-alarms`, and then press Enter.
2. In the list, locate the alarm you want to edit.
3. At the command prompt, type `aws cloudwatch put-metric-alarm --alarm-name <alarm name>`, where `<alarm_name>` is the name of the alarm you want to edit. Specify all of the parameters in the alarm you want to edit, and change any of the values as appropriate.

Checking Alarm Status

You can check the status of your billing alarms using the Amazon CloudWatch console or the AWS command line interface (CLI).

To check alarm status using the Amazon CloudWatch console

1. Sign in to the AWS Management Console and 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 the US East (N. Virginia) region and represent worldwide charges.

From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **Alarms**, click **Billing**.

To check alarm status using the CLI

- At a command prompt, type `aws cloudwatch describe-alarms`, press Enter, and then in the list, locate the AWS/Billing alarm you want to check.

Deleting a Billing Alarm

You can delete a billing alarm when you no longer need it using the Amazon CloudWatch console or the AWS command line interface (CLI).

To delete a billing alarm using the Amazon CloudWatch console

1. Sign in to the AWS Management Console and 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 the US East (N. Virginia) region and represent worldwide charges.

From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#).



3. In the navigation pane, under **Alarms**, click **Billing**.
4. In the list of alarms, select the check box next to the alarm you want to delete, and then click **Delete**.
5. In the **Delete Alarms** dialog box, click **Yes, Delete**.

To delete a billing alarm using the CLI

1. At a command prompt, type `aws cloudwatch describe-alarms`, and then press Enter.
2. In the list, locate the alarm you want to delete.
3. At the command prompt, type `aws cloudwatch delete-alarms --alarm-names <alarm_name>`, where `<alarm_name>` is the name of the alarm you want to delete.
4. At the `Are you sure you want to delete these Alarms?` prompt, type `Y`.

Monitoring Log Files

You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS CloudTrail, or other sources. You can then retrieve the associated log data from CloudWatch Logs using the Amazon CloudWatch console, the CloudWatch Logs commands in the AWS CLI, or the CloudWatch Logs SDK.

You can use CloudWatch Logs to:

- **Monitor Logs from Amazon EC2 Instances in Real-time**—You can use CloudWatch Logs to monitor applications and systems using log data. For example, CloudWatch Logs can track the number of errors that occur in your application logs and send you a notification whenever the rate of errors exceeds a threshold you specify. CloudWatch Logs uses your log data for monitoring; so, no code changes are required. For example, you can monitor application logs for specific literal terms (such as "NullReferenceException") or count the number of occurrences of a literal term at a particular position in log data (such as "404" status codes in an Apache access log). When the term you are searching for is found, CloudWatch Logs reports the data to a CloudWatch metric that you specify.

To get started with CloudWatch Logs on an Amazon EC2 instance running Linux, see [Getting Started with CloudWatch Logs \(p. 114\)](#).

To get started with CloudWatch Logs on an Amazon EC2 instance running Microsoft Windows, see [Sending Performance Counters to CloudWatch and Logs to CloudWatch Logs](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.

- **Monitor AWS CloudTrail Logged Events**—You can create alarms in CloudWatch and receive notifications of particular API activity as captured by CloudTrail and use the notification to perform troubleshooting.

To get started with CloudWatch Logs and logged events in CloudTrail, see [Sending CloudTrail Events to CloudWatch Logs](#) in the *AWS CloudTrail User Guide*.

- **Archive Log Data**—You can use CloudWatch Logs to store your log data in highly durable storage. You can change the log retention setting so that any log events older than this setting are automatically deleted. The CloudWatch Logs agent makes it easy to quickly send both rotated and non-rotated log data off of a host and into the log service. You can then access the raw log data when you need it.

Concepts

The terminology and concepts that are central to your understanding and use of CloudWatch Logs are described below.

Log Events

A log event is a record of some activity recorded by the application or resource being monitored. The log event record that CloudWatch Logs understands contains two properties: the timestamp of when the event occurred, and the raw event message. Event messages must be UTF-8 encoded.

Log Streams

A log stream is a sequence of log events that share the same source. More specifically, a log stream is generally intended to represent the sequence of events coming from the application instance or resource being monitored. For example, a log stream may be associated with an Apache access log on a specific host. Empty log streams are automatically deleted after two months after all data in the log stream has expired. For example, you could create a log stream today with a log retention of two months. If you don't send any log data to that log stream, after two months has elapsed, that log stream is automatically deleted.

Log Groups

Log groups define groups of log streams that share the same retention, monitoring, and access control settings. Each log stream has to belong to one log group. For example, a typical log group organization for a fleet of Apache web servers could be the following: `MyWebsite.com/Apache/access_log`, or `MyWebsite.com/Apache/error_log`.

Metric Filters

Metric filters can be used to express how the service would extract metric observations from ingested events and transform them to data points in a CloudWatch metric. Metric filters are assigned to log groups, and all of the filters assigned to a log group are applied to their log streams.

Retention Settings

Retention settings can be used to specify how long log events are kept in CloudWatch Logs. Expired log events get deleted automatically. Just like metric filters, retention settings are also assigned to log groups, and the retention assigned to a log group is applied to their log streams.

Getting Started with CloudWatch Logs

You can publish log data from Amazon EC2 instances running Linux or Windows Server, and logged events from AWS CloudTrail. CloudWatch Logs can consume logs from resources in any region, but you can only view the log data in the CloudWatch console in the US East (N. Virginia) region, US West (Oregon) region, or EU (Ireland) region.

This section describes the steps necessary for installing the CloudWatch Logs agent on an Amazon EC2 instance running Amazon Linux or Ubuntu Server.

To get started with CloudWatch Logs on an Amazon EC2 instance running Microsoft Windows, see [Sending Performance Counters to CloudWatch and Logs to CloudWatch Logs](#) in the *Amazon EC2 User Guide for Microsoft Windows Instances*.

To get started with CloudWatch Logs and logged events in CloudTrail, see [Sending CloudTrail Events to CloudWatch Logs](#) in the *AWS CloudTrail User Guide*.

CloudWatch Logs Agent Prerequisites

The CloudWatch Logs agent requires the following versions of Python and Linux:

- Python version 2.6, 2.7, 3.0, or 3.3
- Amazon Linux version 2014.03.02
- Ubuntu Server version 12.04, or 14.04
- CentOS version 6, 6.3, 6.4, or 6.5
- Red Hat Enterprise Linux (RHEL) version 6.5 or 7.0

Topics

- Quick Start: Install and Configure the CloudWatch Logs Agent on an Existing EC2 Instance (p. 115)
- Quick Start: Install and Configure the CloudWatch Logs Agent on a New EC2 Instance (p. 118)
- Quick Start: Install the CloudWatch Logs Agent Using AWS OpsWorks and Chef (p. 121)
- Quick Start: Install the CloudWatch Logs Agent Using AWS CloudFormation (p. 125)
- Report the CloudWatch Logs Agent's Status (p. 126)
- Start the CloudWatch Logs Agent (p. 126)
- Stop the CloudWatch Logs Agent (p. 126)
- CloudWatch Logs Agent Reference (p. 127)

Quick Start: Install and Configure the CloudWatch Logs Agent on an Existing EC2 Instance

You can use the Amazon CloudWatch Logs agent installer on an existing EC2 instance to install and configure the CloudWatch Logs agent. After installation is complete, the agent confirms that it has started and it stays running until you disable it.

In addition to the agent, you can also publish log data using the AWS CLI, CloudWatch Logs SDK, or the CloudWatch Logs API. The AWS CLI is best suited for publishing data at the command line or through scripts. The CloudWatch Logs SDK is best suited for publishing log data directly from applications or building your own log publishing application.

Step 1: Configure your IAM role or user for CloudWatch Logs

The CloudWatch Logs agent supports IAM roles and users. If your instance already has an IAM role associated with it, make sure that you include the IAM policy below. If you don't already have an IAM role assigned to your instance, you'll need to use your IAM or root user credentials for the next steps because you cannot assign an IAM role to an existing instance; you can only specify a role when you launch a new instance.

For more information about IAM users and policies, see [IAM Users and Groups](#) and [Managing IAM Policies](#) in *Using IAM*.

To configure your IAM role or user for CloudWatch Logs

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, click **Roles**, and then in the **Role Name** column, click an IAM role.
3. Click **Permissions**, and then click **Attach Role Policy**.
4. On the **Set Permissions** page, click **Custom Policy**, and then click **Select**.

For more information about creating custom policies, see [IAM Policies for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. On the second **Set Permissions** page, in the **Policy Name** field, type a name for the policy.
6. In the **Policy Document** field, paste in the following policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [
```

```
        "logs: *"
    ],
"Resource": [
    "arn:aws:logs: * : * : *"
]
}
}
```

7. Click **Apply Policy**.

Step 2: Install and configure CloudWatch Logs on an existing Amazon EC2 instance

The process for installing the CloudWatch Logs agent differs depending on whether your Amazon EC2 instance is running Amazon Linux, Ubuntu, CentOS, or Red Hat. Use the steps appropriate for the version of Linux on your instance.

To install and configure CloudWatch Logs on an existing Amazon Linux instance

Starting with Amazon Linux AMI 2014.09, the CloudWatch Logs agent is available as an RPM installation with the `awslogs` package. Earlier versions of Amazon Linux can access the `awslogs` package by updating their instance with the `sudo yum update -y` command. By installing the `awslogs` package as an RPM instead of the using the CloudWatch Logs installer, your instance will receive regular package updates and patches from Amazon without having to manually reinstall the CloudWatch Logs agent.

Caution

Do not update the CloudWatch Logs agent using the RPM installation method if you previously used the Python script to install the agent. Doing so may cause configuration issues that prevent the CloudWatch Logs agent from sending your logs to CloudWatch.

1. Connect to your Amazon Linux instance. For more information, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you have trouble connecting, see [Troubleshooting Connecting to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

2. Update your Amazon Linux instance to pick up the latest changes in the package repositories.

```
[ec2-user ~]$ sudo yum update -y
```

3. Install the `awslogs` package.

```
[ec2-user ~]$ sudo yum install -y awslogs
```

4. Edit the `/etc/awslogs/awscli.conf` file and in the `[default]` section, specify the region where you want to view log data and add your credentials.

```
region = <us-east-1, us-west-2, or eu-west-1>
aws_access_key_id = <YOUR ACCESS KEY>
aws_secret_access_key = <YOUR SECRET KEY>
```

Note

Adding your credentials here is optional if your instance was launched using an IAM role or user with the appropriate permissions to use CloudWatch Logs.

5. Edit the `/etc/awslogs/awslogs.conf` file to configure the logs you would like to track. For more information on editing this file, see [CloudWatch Logs Agent Reference \(p. 127\)](#).
6. Start the `awslogs` service.

```
[ec2-user ~]$ sudo service awslogs start  
Starting awslogs: [ OK ]
```

7. (Optional) Check the `/var/log/awslogs.log` file for errors logged when starting the service.
8. (Optional) Run the following command to start the `awslogs` service at each system boot.

```
[ec2-user ~]$ sudo chkconfig awslogs on
```

9. You should see the newly created log group and log stream in the CloudWatch console after the agent has been running for a few moments.

To view your logs, see [Viewing Log Data \(p. 131\)](#).

To install and configure CloudWatch Logs on an existing Ubuntu Server, CentOS, or Red Hat instance

If you're using an AMI running Ubuntu Server, CentOS, or Red Hat, use the following procedure to manually install the CloudWatch Logs agent on your instance.

1. Connect to your EC2 instance. For more information, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you have trouble connecting, see [Troubleshooting Connecting to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

2. Run the CloudWatch Logs agent installer. On the instance, open a command prompt, type the following commands, and then follow the prompts.

Note

On Ubuntu, run `apt-get update` before running the commands below.

```
wget https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py
```

```
sudo python ./awslogs-agent-setup.py --region us-east-1
```

Note

You can install the CloudWatch Logs agent by specifying the `us-east-1`, `us-west-2`, or `eu-west-1` regions.

The CloudWatch Logs agent installer requires certain information during set up. Before you start, you will need to know what log file you want to monitor and its timestamp format. You should also have the following information ready:

Item	Description
AWS Access Key ID	Press enter if using an IAM role. Otherwise, enter your AWS access key ID.
AWS Secret Access Key	Press enter if using an IAM role. Otherwise, enter your AWS secret access key.
Default region name	Press enter. The default is us-east-1. You can set this to us-east-1, us-west-2, or eu-west-1.
Default output format	Leave blank and press enter.
Path of log file to upload	The location of the file that contains the log data you want to send. The installer will suggest a path for you.
Destination Log Group name	The name for your log group. The installer will suggest a log group name for you.
Destination Log Stream name	By default, this is the name of the host. The installer will suggest a host name for you.
Timestamp format	Specify the format of the timestamp within the specified log file. Choose custom to specify your own format.
Initial position	How data will be uploaded. Set this to start_of_file to upload everything in the data file. Set to end_of_file to upload only newly appended data.

After you have completed these steps, the installer asks if you want to configure another log file. You can run the process as many times as you like for each log file. If you have no more log files to monitor, choose **N** when prompted by the installer to set up another log. For more information about the settings in the agent configuration file, see [CloudWatch Logs Agent Reference \(p. 127\)](#).

Note

Configuring multiple log sources to send data to a single log stream is not supported.

3. You should see the newly created log group and log stream in the CloudWatch console after the agent has been running for a few moments.

To view your logs, see [Viewing Log Data \(p. 131\)](#).

Quick Start: Install and Configure the CloudWatch Logs Agent on a New EC2 Instance

You can use Amazon EC2 user data, a feature of Amazon EC2 that allows parametric information to be passed to the instance on launch, to install and configure the CloudWatch Logs agent on that instance. To pass the CloudWatch Logs agent installation and configuration information to Amazon EC2, you can provide the configuration file in a network location such as an Amazon S3 bucket. You can launch a new Amazon EC2 instance and enable logs by performing the following steps:

To launch a new instance and enable CloudWatch Logs

1. Create an agent configuration file that describes all your log groups and log streams:

Sample agent configuration file for Amazon Linux

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

Sample agent configuration file for Ubuntu

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/syslog]
file = /var/log/syslog
log_group_name = /var/log/syslog
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

The agent configuration file describes the log files to monitor and the target log groups and log streams to upload it to. The agent consumes this configuration file and starts monitoring/uploading all the log files described in it. For more information about the settings in the agent configuration file, see [CloudWatch Logs Agent Reference \(p. 127\)](#).

Save it as a text file (for example, awslogs.cfg) either on the AMI's filesystem, in a publicly accessible http/https location, or an Amazon S3 location (for example, s3://myawsbucket/my-config-file). For more information about assigning permissions to an Amazon S3 bucket, see [Specifying Resources in a Policy](#) in the *Amazon Simple Storage Service Developer Guide*.

Note

Configuring multiple log sources to send data to a single log stream is not supported.

2. Open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, click **Roles**, and then in the contents pane, click **Create New Role**.
4. On the **Set Role Name** page, enter a name for the role and click **Next Step**.
5. On the **Select Role Type** page, under **AWS Service Roles**, click **Select** next to Amazon EC2.
6. On the **Set Permissions** page, click **Custom Policy**, and then click **Select**. For more information about creating custom policies, see [IAM Policies for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.
7. On the second **Set Permissions** page, in the **Policy Name** field, type a name for the policy.
8. In the **Policy Document** field, paste in the following policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:*",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*",
                "arn:aws:s3:::myawsbucket/*"
```

```
        ]  
    }  
}  
}
```

9. Click **Next Step**, and then review the role information. If you're satisfied with the role, click **Create Role**.

For more information about IAM users and policies, see [IAM Users and Groups](#) and [Managing IAM Policies in Using IAM](#).

10. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
11. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



12. On the Amazon EC2 console dashboard, click **Launch Instance**.

For more information about how to launch an instance, see [Launching an Instance](#) in *Amazon EC2 User Guide for Linux Instances*.

13. On the **Step 1: Choose an Amazon Machine Image (AMI)** page, select the Linux instance type you want to launch, and then on the **Step 2: Choose an Instance Type** page, click **Next: Configure Instance Details**.

Note

Make sure that [cloud-init](http://cloudinit.readthedocs.org/en/latest/index.html) (<http://cloudinit.readthedocs.org/en/latest/index.html>) is installed on your Amazon Machine Image (AMI). Amazon Linux, Ubuntu, and RHEL instances already include cloud-init, but CentOS and other AMIs in the AWS Marketplace may not.

14. On the **Step 3: Configure Instance Details** page, In the **IAM role** field, select the IAM role that you created above.
15. Under **Advanced Details**, in the **User data** field, paste in the script and update the **-c** option with the location of the configuration file:

```
#!/bin/bash  
wget https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-setup.py  
chmod +x ./awslogs-agent-setup.py  
../awslogs-agent-setup.py -n -r us-east-1 -c s3://myawsbucket/my-config-file
```

Note

You can install the CloudWatch Logs agent by specifying the us-east-1, us-west-2, or eu-west-1 regions.

16. Make any other changes to the instance that you want, review your launch settings, and then click **Launch**.
17. You should see the newly created log group and log stream in the CloudWatch console after the agent has been running for a few moments.

To view your logs, see [Viewing Log Data \(p. 131\)](#).

Quick Start: Install the CloudWatch Logs Agent Using AWS OpsWorks and Chef

You can install the CloudWatch Logs agent and create log streams using AWS OpsWorks and Chef, which is a third-party systems and cloud infrastructure automation tool. Chef uses "recipes," which you write to install and configure software on your computer, and "cookbooks," which are collections of recipes, to perform its configuration and policy distribution tasks. For more information, see [Chef](#).

The Chef recipes examples below show how to monitor one log file on each EC2 instance. The recipes use the stack name as the log group and the instance's hostname as the log stream name. If you want to monitor multiple log files, you need to extend the recipes to create multiple log groups and log streams.

Topics

- [Step 1: Create Custom Recipes \(p. 121\)](#)
- [Step 2: Create an AWS OpsWorks Stack \(p. 123\)](#)
- [Step 3: Extend Your IAM Role \(p. 123\)](#)
- [Step 4: Add a Layer \(p. 124\)](#)
- [Step 5: Add an Instance \(p. 124\)](#)
- [Step 6: View Your Logs \(p. 124\)](#)

Step 1: Create Custom Recipes

Create a repository to store your recipes. AWS OpsWorks supports Git and Subversion, or you can store an archive in Amazon S3. The structure of your cookbook repository is described in [Cookbook Repositories](#) in the *AWS OpsWorks User Guide*. The examples below assume that the cookbook is named `logs`. The `install.rb` recipe installs the CloudWatch Logs agent. You can also download the cookbook example ([CloudWatchLogs-Cookbooks.zip](#)).

Create a file named `metadata.rb` that contains the following code:

```
#metadata.rb

name          'logs'
version       '0.0.1'
```

Create the CloudWatch Logs configuration file:

```
#config.rb

template "/tmp/cwlogs.cfg" do
  cookbook "logs"
  source  "cwlogs.cfg.erb"
  owner   "root"
  group   "root"
```

```
mode 0644  
end
```

Download and install the CloudWatch Logs agent:

```
# install.rb

directory "/opt/aws/cloudwatch" do
  recursive true
end

remote_file "/opt/aws/cloudwatch/awslogs-agent-setup.py" do
  source "https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-
agent-setup.py"
  mode "0755"
end

execute "Install CloudWatch Logs agent" do
  command "/opt/aws/cloudwatch/awslogs-agent-setup.py -n -r us-east-1 -c
/tmp/cwlogs.cfg"
  not_if { system "pgrep -f aws-logs-agent-setup" }
end
```

Note

You can install the CloudWatch Logs agent by specifying the us-east-1, us-west-2, or eu-west-1 regions.

This recipe uses a cwlogs.cfg.erb template file that you can modify to specify various attributes such as what files to log. For more information about these attributes, see [CloudWatch Logs Agent Reference \(p. 127\)](#).

```
[general]
# Path to the AWSLogs agent's state file. Agent uses this file to maintain
# client side state across its executions.
state_file = /var/awslogs/state/agent-state

## Each log file is defined in its own section. The section name doesn't
## matter as long as its unique within this file.
#
#[kern.log]
#
## Path of log file for the agent to monitor and upload.
#
## Name of the destination log group.
#
## Name of the destination log stream.
#
## Format specifier for timestamp parsing.
#
```

```
#datetime_format = %b %d %H:%M:%S
#
#
[<%= node[:opsworks][:stack][:name] %>]
datetime_format = [%Y-%m-%d %H:%M:%S]
log_group_name = <%= node[:opsworks][:stack][:name].gsub(' ', '_') %>
file = <%= node[:cwlogs][:logfile] %>
log_stream_name = <%= node[:opsworks][:instance][:hostname] %>
```

The template gets the stack name and host name by referencing the corresponding attributes in the stack configuration and deployment JSON. The attribute that specifies the file that you want to log is defined in the cwlogs cookbook's default.rb attributes file (logs/attributes/default.rb).

```
default[:cwlogs][:logfile] = '/var/log/aws/opsworks/opsworks-agent.statistics.log'
```

Step 2: Create an AWS OpsWorks Stack

1. Open the AWS OpsWorks console at <https://console.aws.amazon.com/opsworks/>.
2. Select **Add a Stack** to create an AWS OpsWorks stack.
3. Give it a name and click **Advanced**.
4. Under **Configuration Management**, set **Use custom Chef Cookbooks** to **Yes**.
5. In the **Repository type** list, select the repository type you use. If you're using the above example, choose **Http Archive**.
6. In the **Repository URL** field, enter the repository where you stored the cookbook that you created in the previous step. If you're using the above example, enter <https://s3.amazonaws.com/aws-cloudwatch/downloads/CloudWatchLogs-Cookbooks.zip>.
7. Click **Add Stack** to create the stack.

Step 3: Extend Your IAM Role

To use CloudWatch Logs with your AWS OpsWorks instances, you need to extend the IAM role used by your instances.

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, click **Roles**, and then in the contents pane, select the instance role used by your AWS OpsWorks stack. You can find the one used by your stack in the stack settings (the default is aws-opsworks-ec2-role).
3. In the lower pane, on the **Permissions** tab, click **Attach Role Policy**.
4. On the **Manage Role Permissions** page, select **Custom Policy**, and then click **Select**.

For more information about creating custom policies, see [IAM Policies for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. On the second **Set Permissions** page, in the **Policy Name** field, type a name for the policy.
6. In the **Policy Document** field, paste in the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
    "Effect": "Allow",
    "Action": [
        "logs:/*"
    ],
    "Resource": [
        "arn:aws:logs:*:*:*"
    ]
}
```

7. Click **Apply Policy**.

Step 4: Add a Layer

1. Open the AWS OpsWorks console at <https://console.aws.amazon.com/opsworks/>.
2. In the navigation pane, click **Layers**.
3. In the contents pane, select a layer, and then click **Add a layer**.
4. In the contents pane, click the new layer, and then click **Recipes**.
5. In the **Custom Chef Recipes** section, there are several headings—*Setup*, *Configure*, *Deploy*, *Undeploy*, and *Shutdown*, which correspond to AWS OpsWorks lifecycle events. AWS OpsWorks triggers these events at these key points in instance's lifecycle, which runs the associated recipes.
6. Enter *logs::config*, *logs::install* next to **Setup**, click **+** to add it to the list, and then click **Save**.

AWS OpsWorks runs this recipe on each of the new instances in this layer, right after the instance boots.

Step 5: Add an Instance

The layer only controls how to configure instances. You now need to add some instances to the layer and start them.

1. Open the AWS OpsWorks console at <https://console.aws.amazon.com/opsworks/>.
2. In the navigation pane, click **Instances**, and then under your layer, click **+ Instance**.
3. Accept the default settings, and then click **Add Instance** to add the instance to the layer.
4. In the row's **Actions** column, click **start** to start the instance.

AWS OpsWorks launches a new EC2 instance and configures CloudWatch Logs. The instance's status changes to online when it's ready.

Step 6: View Your Logs

You should see the newly created log group and log stream in the CloudWatch console after the agent has been running for a few moments.

To view your logs, see [Viewing Log Data \(p. 131\)](#).

Quick Start: Install the CloudWatch Logs Agent Using AWS CloudFormation

AWS CloudFormation enables you to describe your AWS resources in JSON-formatted templates. With AWS CloudFormation, you can describe and then quickly and consistently provision log groups and metric filters in Amazon CloudWatch Logs. You can also use AWS CloudFormation to install and configure the CloudWatch Logs agent on Amazon EC2 instances. For example, if you have multiple Apache web servers on Amazon EC2 instances, you can write a single AWS CloudFormation template that defines the web server logs and the information from those logs that you want to monitor. You can then reuse the template for all of your Apache web servers. For more information about AWS CloudFormation, see the [Introduction](#) in the *AWS CloudFormation User Guide*.

For more information about CloudWatch resources in AWS CloudFormation, see the [AWS::Logs::LogGroup](#) and [AWS::Logs::MetricFilter](#) in the *AWS CloudFormation User Guide*.

The following template snippet creates a log group and metric filter. The log group retains log events for 7 days. The metric filter counts the number of 404 occurrences. It sends a metric value of 1 each time the status code field equals 404.

Example

```
"WebServerLogGroup": {  
    "Type": "AWS::Logs::LogGroup",  
    "Properties": {  
        "RetentionInDays": 7  
    }  
},  
  
"404MetricFilter": {  
    "Type": "AWS::Logs::MetricFilter",  
    "Properties": {  
        "LogGroupName": {  
            "Ref": "WebServerLogGroup"  
        },  
        "FilterPattern": "[ip, identity, user_id, timestamp, request, status_code  
= 404, size, ...]",  
        "MetricTransformations": [  
            {  
                "MetricValue": "1",  
                "MetricNamespace": "test/404s",  
                "MetricName": "test404Count"  
            }  
        ]  
    }  
}
```

For a stack named **MyStack**, the previous example creates a log group named **MyStack-my-LogGroup-unique hash** and a metric filter named **myMetricFilter-unique hash**. For a complete sample template that includes an Amazon EC2 instance and Amazon CloudWatch alarms, see [Amazon CloudWatch Logs Sample](#) in the *AWS CloudFormation User Guide*.

Report the CloudWatch Logs Agent's Status

To report the agent's status

1. Connect to your EC2 instance. For more information, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
If you have trouble connecting, see [Troubleshooting Connecting to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*
2. At a command prompt, type the following command:

```
sudo service awslogs status
```

3. Check the **/var/log/awslogs.log** file for any errors, warnings, or issues with the CloudWatch Logs agent.

Start the CloudWatch Logs Agent

If the CloudWatch Logs agent did not start automatically after installation, or if you stopped the agent, you can manually start it.

To start the agent

1. Connect to your EC2 instance. For more information, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
If you have trouble connecting, see [Troubleshooting Connecting to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. At a command prompt, type the following command:

```
sudo service awslogs start
```

Stop the CloudWatch Logs Agent

To stop the agent

1. Connect to your EC2 instance. For more information, see [Connect to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
If you have trouble connecting, see [Troubleshooting Connecting to Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. At a command prompt, type the following command:

```
sudo service awslogs stop
```

CloudWatch Logs Agent Reference

The CloudWatch Logs agent provides an automated way to send log data to CloudWatch Logs for Amazon EC2 instances running Amazon Linux or Ubuntu. The agent is comprised of the following components:

- A plug-in to the AWS CLI that pushes log data to CloudWatch Logs.
- A script (daemon) that runs the CloudWatch Logs aws logs push command to send data to CloudWatch Logs.
- A cron job ensures that the daemon is always running.

Agent Configuration File

The CloudWatch Logs agent configuration file describes information needed by aws logs push command. The agent configuration file's [general] section defines common configurations that apply to all log streams. The [logstream] section defines the information necessary to send a local file to a remote log stream. You can have more than one [logstream] section, but each must have a unique name within the configuration file, e.g., [logstream1], [logstream2], and so on. The [logstream] value along with the first line of data in the log file, define the log file's identity.

```
[general]
state_file = <value>

[logstream1]
log_group_name = <value>
log_stream_name = <value>
datetime_format = <value>
time_zone = [LOCAL|UTC]
file = <value>
file_fingerprint_lines = <integer> | <integer-integer>
multi_line_start_pattern = <regex> | {datetime_format}
initial_position = [start_of_file|end_of_file]
encoding = [ascii|utf_8|...]
buffer_duration = <integer>

[logstream2]
...
```

state_file

Specifies where the state file is stored.

log_group_name

Specifies the destination log group. A log group will be created automatically if it doesn't already exist.

log_stream_name

Specifies the destination log stream. You can use a literal string or predefined variables ({instance_id}, {hostname}, {ip_address}), or combination of both to define a log stream name. A log stream will be created automatically if it doesn't already exist.

datetime_format

Specifies how the timestamp is extracted from logs. The timestamp is used for retrieving log events and generating metrics. The current time is used for each log event if the **datetime_format** isn't provided. If the provided **datetime_format** is invalid for a given log message, the timestamp from the last log event with a successfully parsed timestamp will be used. If no previous log events exist, the current time is used.

The common **datetime_format** codes are listed below. You can also use any **datetime_format** codes supported by Python, `datetime.strptime()`. The timezone offset (%z) is also supported even though

it's not supported until python 3.2, [+]-HHMM without colon(:). For more information, see [strftime\(\) and strftime\(\) Behavior](#).

%y: Year without century as a zero-padded decimal number. 00, 01, ..., 99

%Y: Year with century as a decimal number. 1970, 1988, 2001, 2013

%b: Month as locale's abbreviated name. Jan, Feb, ..., Dec (en_US);

%B: Month as locale's full name. January, February, ..., December (en_US);

%m: Month as a zero-padded decimal number. 01, 02, ..., 12

%d: Day of the month as a zero-padded decimal number. 01, 02, ..., 31

%H: Hour (24-hour clock) as a zero-padded decimal number. 00, 01, ..., 23

%I: Hour (12-hour clock) as a zero-padded decimal number. 01, 02, ..., 12

%p: Locale's equivalent of either AM or PM.

%M: Minute as a zero-padded decimal number. 00, 01, ..., 59

%S: Second as a zero-padded decimal number. 00, 01, ..., 59

%f: Microsecond as a decimal number, zero-padded on the left. 000000, ..., 999999

%z: UTC offset in the form +HHMM or -HHMM. +0000, -0400, +1030

Example formats:

Syslog: '%b %d %H:%M:%S', e.g. Jan 23 20:59:29

Log4j: '%d %b %Y %H:%M:%S', e.g. 24 Jan 2014 05:00:00

ISO8601: '%Y-%m-%dT%H:%M:%S%z', e.g. 2014-02-20T05:20:20+0000

time_zone

Specifies the time zone of log event timestamp. The two supported values are UTC and LOCAL. The default is LOCAL, which is used if time zone can't be inferred based on **datetime_format**.

file

Specifies log files that you want to push to CloudWatch Logs. File can point to a specific file or multiple files (using wildcards such as /var/log/system.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.2014-06-01-01, access_log.2014-06-01-02, and so on, but not multiple different kinds of files, such as access_log_80 and access_log_443. To specify multiple different kinds of files, add another log stream entry to the configuration file so each kind of log files goes to different log group. Zipped files are not supported.

file_fingerprint_lines

Specifies the range of lines for identifying a file. The valid values are one number or two dash delimited numbers, such as '1', '2-5'. The default value is '1' so the first line is used to calculate fingerprint. Fingerprint lines are not sent to CloudWatch Logs unless all the specified lines are available.

multi_line_start_pattern

Specifies the pattern for identifying the start of a log message. A log message is made of a line that matches the pattern and any following lines that don't match the pattern. The valid values are regular expression or {datetime_format}. When using {datetime_format}, the datetime_format option should be specified. The default value is '^[\s]' so any line that begins with non-whitespace character closes the previous log message and starts a new log message.

initial_position

Specifies where to start to read data (start_of_file or end_of_file). The default is start_of_file. It's only used if there is no state persisted for that log stream.

encoding

Specifies the encoding of the log file so that the file can be read correctly. The default is utf_8. Encodings supported by Python codecs.decode() can be used here.

Caution

Specifying an incorrect encoding might cause data loss because characters that cannot be decoded will be replaced with some other character.

Below are some common encodings:

```
ascii, big5, big5hkscs, cp037, cp424, cp437, cp500, cp720, cp737, cp775, cp850, cp852, cp855, cp856, cp857, cp858, cp860, cp861, cp862, cp863, cp864, cp865, cp866, cp869, cp874, cp875, cp932, cp949, cp950, cp1006, cp1026, cp1140, cp1250, cp1251, cp1252, cp1253, cp1254, cp1255, cp1256, cp1257, cp1258, euc_jp, euc_jis_2004, euc_jisx0213, euc_kr, gbk, gb2312, gbk, gb18030, hz, iso2022_jp, iso2022_jp_1, iso2022_jp_2, iso2022_jp_2004, iso2022_jp_3, iso2022_jp_ext, iso2022_kr, latin_1, iso8859_2, iso8859_3, iso8859_4, iso8859_5, iso8859_6, iso8859_7, iso8859_8, iso8859_9, iso8859_10, iso8859_13, iso8859_14, iso8859_15, iso8859_16, johab, koi8_r, koi8_u, mac_cyrillic, mac_greek, mac_iceland, mac_latin2, mac_roman, mac_turkish, ptcp154, shift_jis, shift_jis_2004, shift_jisx0213, utf_32, utf_32_be, utf_32_le, utf_16, utf_16_be, utf_16_le, utf_7, utf_8, utf_8_sig
```

buffer_duration

Specifies the time duration for the batching of log events. The minimum value is 5000ms and default value is 5000ms.

CloudWatch Logs Agent FAQs

What kinds of file rotations are supported?

The following file rotation mechanisms are supported:

1. Renaming existing log files with a numerical suffix, then recreating the original empty log file. For example, /var/log/syslog.log is renamed /var/log/syslog.log.1. If /var/log/syslog.log.1 already exists from a previous rotation, it is renamed /var/log/syslog.log.2.
2. Truncating the original log file in place after creating a copy. For example, /var/log/syslog.log is copied to /var/log/syslog.log.1 and /var/log/syslog.log is truncated. There might be data loss for this case, so be careful about using this file rotation mechanism.
3. Creating a new file with a common pattern as the old one. For example, /var/log/syslog.log.2014-01-01 remains there and /var/log/syslog.log.2014-01-02 is created.

The fingerprint (source id) of the file is calculated by hashing the log stream key and the first line of file content. To override this behavior, the **file_fingerprint_lines** option can be used. When file rotation happens, the new file is supposed to have new content and the old file is supposed to not have content appended, agent will push the new file once it finishes reading the old file.

How are log entries converted to log events?

Log events contain two properties: the timestamp of when the event occurred, and the raw log message. By default, any line that begins with non-whitespace character closes the previous log message if there is one, and starts a new log message. To override this behavior, the **multi_line_start_pattern** can be used and any line that matches the pattern starts a new log message. The pattern could be any regex or '{datetime_format}'. For example, if the first line of every log message contains timestamp like '2014-01-02T13:13:01Z', then the **multi_line_start_pattern** can be set to '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}Z'. To simplify the configuration, the '{datetime_format}' variable can be used if the **datetime_format option** is specified. For the same example, if **datetime_format** is set to '%Y-%m-%dT:%M:%S%z', then **multi_line_start_pattern** could be simply '{datetime_format}'.

The current time is used for each log event if the **datetime_format** isn't provided. If the provided **datetime_format** is invalid for a given log message, the timestamp from the last log event with a successfully parsed timestamp will be used. If no previous log events exist, the current time is used. A warning message is logged when a log event falls back to current time or time of previous log event.

Timestamps are used for retrieving log events and generating metrics, so if you specify the wrong format, log events could become non-retrievable and generate wrong metrics.

How are log events batched?

- A batch becomes full and will be published when any of the following conditions are met:
1. The **buffer_duration** amount of time has passed since the first log event is added.
 2. Less than 32KB of log events have been accumulated but adding the new log event exceeds the 32KB constraint.
 3. 1000 log events have been accumulated.
 4. Log events from the batch don't span more than 24 hours but adding the new log event exceeds the 24 hours constraint.

What would cause log entries, log events, or batches to be skipped or truncated?

To follow the constraint of the PutLogEvents API, the following issues could cause a log event or batch to be skipped.

Note

The CloudWatch Logs agent writes a warning to its log when data is skipped.

1. If the size of a log event exceeds 32KB, the log event will be skipped completely.
2. If the timestamp of log event is more than 2 hours in future, the log event is skipped.
3. If the timestamp of log event is more than 14 days in past, the log event is skipped.
4. If any log event is older than the retention period of log group, the whole batch is skipped.

Does stopping the agent cause data loss/duplicates?

Not as long as the state file is available and no file rotation has happened since the last run. The CloudWatch Logs agent can start from where it stopped and continue pushing the log data.

Can I point different log files from the same or different hosts to the same log stream?

Configuring multiple log sources to send data to a single log stream is not supported.

What API calls does the agent make (or what actions should I add to my IAM policy)?

The CloudWatch Logs agent requires the CreateLogGroup, CreateLogStream, DescribeLogStreams and PutLogEvents actions. See the sample IAM policy below.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:PutLogEvents",  
                "logs:DescribeLogStreams"  
            ],  
            "Resource": [  
                "arn:aws:logs:*:*:  
            ]  
        }  
    ]  
}
```

I don't want the CloudWatch Logs agent to create log groups or log streams automatically. How can I create and delete them and prevent the agent from recreating them?

In your IAM policy, you can restrict the agent to only the following actions: `DescribeLogStreams`, `PutLogEvents`.

Viewing Log Data

You can view the log data on a stream-by-stream basis as sent to CloudWatch Logs by the CloudWatch Logs agent.

To view log data

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, in the **Log Groups** column, click the log group for which you want to view data.
5. In the **Log Groups > Streams for** pane, in the **Log Streams** column, click the log stream's name to view the log data.

Changing Log Retention

By default, log data is stored indefinitely. However, you can configure how long you want to store log data in a log group. Any data older than the current retention setting is automatically deleted. You can change the log retention for each log group at any time.

To change the logs retention setting

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, in the **Expire Events After** column, click the retention setting you want to change.
5. In the **Edit Retention** dialog box, in the **New Retention** list, select the desired log retention, and then click **OK**.

Monitoring Log Data

After the CloudWatch Logs agent begins publishing log data to Amazon CloudWatch, you can begin monitoring the log data by creating one or more metric filters. Metric filters define the terms and patterns to look for in log data as it is sent to CloudWatch Logs. CloudWatch Logs uses these metric filters to turn log data into CloudWatch metrics that you can graph or set an alarm on. Each metric filter is made up of the following key elements:

filter pattern

A symbolic description of how CloudWatch Logs should interpret the data in each log event. For example, a log entry may contain timestamps, IP addresses, strings, and so on. You use the pattern to specify what to look for in the log file.

metric name

The name of the CloudWatch metric to which the monitored log information should be published. For example, you may publish to a metric called ErrorCount.

metric namespace

The destination namespace of the new CloudWatch metric.

metric value

What to publish to the metric. For example, if you're counting the occurrences of a particular term like "Error", the value will be "1" for each occurrence. If you're counting the bytes transferred the published value will be the value in the log event.

Filter and Pattern Syntax

You use metric filters to search for and match terms, phrases, or values in your log events. When a metric filter finds one of the terms, phrases, or values in your log events, it counts each occurrence in a CloudWatch metric. For example, you can create a metric filter to search for and count the occurrence of the word *ERROR* in your log events. Metric filters can also extract values from space-delimited log events, such as the latency of web requests. You can also use conditional operators and wildcards to create exact matches. Before you create a metric filter, you can test your search patterns in the CloudWatch console. The following sections explain the metric filter syntax in more detail.

Matching Terms in Log Events

To search for a term such as *ERROR* in your log events, you would use *ERROR* as your metric filter pattern. You can specify multiple terms in a metric filter pattern, but all terms must appear in your log events for there to be a match. For example, the filter pattern *ERROR* would match the following log event messages:

- [ERROR] A fatal exception has occurred
- Exiting with ERRORCODE: -1

The filter pattern *ERROR Exception* would match the following log event messages:

- [ERROR] Caught IllegalArgumentException
- [ERROR] Unhandled Exception

The filter pattern *"Failed to process the request"* would match the following log event messages:

- [WARN] Failed to process the request
- [ERROR] Unable to continue: Failed to process the request

Note

Metric filter terms that include characters other than alphanumeric or underscore must be placed inside double quotes (""). Metric filters are case sensitive.

Matching Terms in JSON Log Events

You can extract values from JSON log events. To extract values from JSON log events, you need to create a string-based metric filter. Strings containing scientific notation are not supported. The items in the JSON log event data must exactly match the metric filter. You might want to create metric filters in JSON log events to indicate whenever:

- A certain event occurs. For example `eventName` is "UpdateTrail".
- The IP is outside a known subnet. For example, `sourcelpAddress` is not in some known subnet range.
- A combination of two or more other conditions are true. For example, the `eventName` is "UpdateTrail" and the `recipientAccountId` is 123456789012.

Using Metric Filters to Extract Values from Space-Delimited Log Events

You can use metric filters to extract values from space-delimited log events. The characters between a pair of square brackets [] or two double quotes ("") are treated as a single field. For example:

```
127.0.0.1 - frank [10/Oct/2000:13:25:15 -0700] "GET /apache_pb.gif HTTP/1.0"
200 1534
127.0.0.1 - frank [10/Oct/2000:13:35:22 -0700] "GET /apache_pb.gif HTTP/1.0"
500 5324
127.0.0.1 - frank [10/Oct/2000:13:50:35 -0700] "GET /apache_pb.gif HTTP/1.0"
200 4355
```

To specify a metric filter pattern that parses space-delimited events, the metric filter pattern has to specify the fields with a name, separated by commas, with the entire pattern enclosed in square brackets. For example: [ip, user, username, timestamp, request, status_code, bytes].

In cases where you don't know the number of fields, you can use shorthand notation using an ellipsis (...). For example:

```
[..., status_code, bytes]  
[ip, user, ..., status_code, bytes]  
[ip, user, ...]
```

You can also add conditions to your fields so that only log events that match all conditions would match the filters. For example:

```
[ip, user, username, timestamp, request, status_code, bytes > 1000]  
[ip, user, username, timestamp, request, status_code = 200, bytes]  
[ip, user, username, timestamp, request, status_code = 4*, bytes]  
[ip, user, username, timestamp, request = *html*, status_code = 4*, bytes]
```

CloudWatch Logs supports both string and numeric conditional fields. For string fields, you can use = or != operators with an asterisk (*).

For numeric fields, you can use the >, <, >=, <=, =, and != operators.

Using Metric Filters to Extract Values from JSON Log Events

You can use metric filters to extract values from JSON log events. The metric filter syntax for JSON log events uses the following format:

```
{ SELECTOR EQUALITY_OPERATOR STRING }
```

The metric filter must be enclosed in curly braces {}, to indicate this is a JSON expression. The metric filter contains the following parts:

SELECTOR

Specifies what JSON property to check. Property selectors always starts with dollar sign (\$), which signifies the root of the JSON. Property selectors are alphanumeric strings that also support '-' and '_' characters. Array elements are denoted with [NUMBER] syntax, and must follow a property. Examples are: \$.eventId, \$.users[0], \$.users[0].id, \$.requestParameters.instanceId.

EQUALITY_OPERATOR

Can be either = or !=.

STRING

A string with or without quotes. You can use the asterisk '*' wildcard character to match any text at, before, or after a search term. For example, *Event will match PutEvent and GetEvent. Event* will match EventId and EventName. Ev*ent will only match the actual string Ev*ent. Strings that consist entirely of alphanumeric characters do not need to be quoted. Strings that have unicode and other characters such as '@', '\$', '\,' etc. must be enclosed in double quotes to be valid.

Metric Filter Examples

```
{  
    "eventType": "UpdateTrail",  
    "sourceIpAddress": "111.111.111.111",  
    "arrayKey": [  
        "value",  
        "another value"  
    ],
```

```

"objectList": [
    {
        "name": "a",
        "id": 1
    },
    {
        "name": "b",
        "id": 2
    }
]
}

```

The above JSON example would be matched by any of these filters:

```
{ $.eventType = "UpdateTrail" }
```

Filter on the event type being UpdateTrail.

```
{ $.sourceIpAddress != 123.123.* }
```

Filter on the IP address being outside the subnet 123.123 prefix.

```
{ $.arrayKey[0] = "value" }
```

Filter on the first entry in arrayKey being "value". If arrayKey is not an array this will be false.

```
{ $.objectList[1].id = 2 }
```

Filter on the second entry in objectList having a property called id = 2. If objectList is not an array this will be false. If the items in objectList are not objects or do not have an id property, this will be false.

Compound Conditions

You can combine multiple conditions into a compound expression using OR (||) and AND(&&). Parenthesis are allowed and the syntax follows standard order of operations () > && > ||.

```

{
    "user": {
        "id": 1,
        "email": "John.Stiles@example.com"
    },
    "users": [
        {
            "id": 2,
            "email": "John.Doe@example.com"
        },
        {
            "id": 3,
            "email": "Jane.Doe@example.com"
        }
    ],
    "actions": [
        "GET",
        "PUT"
    ]
}

```

```
    "PUT",
    "DELETE"
],
"coordinates": [
    [0, 1, 2],
    [4, 5, 6],
    [7, 8, 9]
]
}
```

Examples

```
{ ($.user.id = 1) && ($.users[0].email = "John.Doe@example.com") }
```

Matches the JSON above.

```
{ ($.user.id = 2 && $.users[0].email = "nonmatch") || $.actions[2] = "GET" }
```

Doesn't match the JSON above.

```
{ $.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = nonmatch
&& $.actions[2] = nomatch }
```

Matches the JSON above.

```
{ ($.user.email = "John.Stiles@example.com" || $.coordinates[0][1] = nonmatch)
&& $.actions[2] = nomatch }
```

Doesn't match the JSON above.

Special Considerations

The SELECTOR must point to a value node (string or number) in the JSON. If it points to an array or object, the filter will not be applied because the log format doesn't match the filter. For example, both `$.users = 1` and `$.users != 1` will fail to match a log event where users is an array:

```
{
    "users": [1, 2, 3]
}
```

Creating Metric Filters

The following examples show how you can create metric filters.

Topics

- [Example: Counting Log Events \(p. 137\)](#)
- [Example: Count occurrences of the word "Error" \(p. 138\)](#)
- [Example: Counting HTTP 404 Responses \(p. 139\)](#)
- [Example: Find and Count 400-level into 4xx Metric \(p. 141\)](#)
- [Example: Extract Bytes Transferred from an Apache Log \(p. 143\)](#)

Example: Counting Log Events

The simplest type of log event monitoring is to count the number of log events that occur. You might want to do this to keep a count of all events, to create a “heartbeat” style monitor or just to practice creating metric filters.

In the following CLI example, a metric filter called `MyAppAccessCount` is applied to the log group `MyApp/access.log` to create the metric `EventCount` in the CloudWatch namespace `YourNamespace`. The filter is configured to match any log event content and to increment the metric by "1".

To create a metric filter using the CloudWatch console

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, select a log group, and then click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, leave the **Filter Pattern** field blank.
6. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** field, enter `MyAppAccessCount`.
7. Under **Metric Details**, in the **Metric Namespace** field, enter `YourNamespace`.
8. In the **Metric Name** field, enter `MyAppAccessEventCount`, and then click **Create Filter**.

To create a metric filter using the AWS CLI

- At a command prompt, type:

```
% aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name MyAppAccessCount \
--filter-pattern '' \
--metric-transformations \
metricName=EventCount,metricNamespace=YourNamespace,metricValue=1
```

You can test this new policy by posting any event data. You should see two data points published to the metric `EventCount`.

To post event data using the AWS CLI

- At a command prompt, type:

```
% aws logs put-log-events \
--log-group-name MyApp/access.log --log-stream-name TestStream1 \
--log-events \
timestamp=1394793518000,message="Test event 1" \
timestamp=1394793518000,message="Test event 2" \
timestamp=1394793528000,message="This message also contains an Error"
```

Example: Count occurrences of the word "Error"

Log events frequently include important messages that you want to count, maybe about the success or failure of operations. For example, an error may occur and be recorded to a log file if a given operation fails. You may want to monitor these entries to understand the trend of your errors.

In the example below, a metric filter is created to monitor for the term Error. The policy has been created and added to the log group MyApp/message.log. CloudWatch Logs publishes a data point to the CloudWatch custom metric ErrorCount in the MyApp/message.log namespace with a value of "1" for every event containing Error. If no event contains the word Error, then no data points are published.

To create a metric filter using the CloudWatch console

- Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
- If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



- In the navigation pane, click **Logs**.
- In the contents pane, select a log group, and then click **Create Metric Filter**.
- On the **Define Logs Metric Filter** screen, in the **Filter Pattern** field, enter `Error`.

Note

All entries in the **Filter Pattern** field are case-sensitive.

- To test your filter pattern, in the **Select Log Data to Test** list, select the log group you want to test the metric filter against, and then click **Test Pattern**.
- Under **Results**, CloudWatch Logs displays a message showing how many occurrences of the filter pattern were found in the log file.

Note

To see detailed results, click **Show test results**.

8. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** field, enter **MyAppErrorCount**.
9. Under **Metric Details**, in the **Metric Namespace** field, enter **YourNameSpace**.
10. In the **Metric Name** field, enter **ErrorCount**, and then click **Create Filter**.

To create a metric filter using the AWS CLI

- At a command prompt, type:

```
% aws logs put-metric-filter \
--log-group-name MyApp/message.log \
--filter-name MyAppErrorCount \
--filter-pattern 'Error' \
--metric-transformations \
metricName=EventCount,metricNamespace=YourNamespace,metricValue=1
```

You can test this new policy by posting events containing the word "Error" in the message.

To post events using the AWS CLI

- At a command prompt, type:

```
% aws logs put-log-events \
--log-group-name MyApp/access.log --log-stream-name TestStream1 \
--log-events \
timestamp=1394793518000,message="This message contains an Error" \
timestamp=1394793528000,message="This message also contains an Error"
```

Note

Patterns are case-sensitive.

Example: Counting HTTP 404 Responses

Using CloudWatch Logs, you can monitor how many times your Apache servers return a HTTP 404 response, which is the response code for page not found. You might want to monitor this to understand how often your site visitors do not find the resource they are looking for. Assume that your log records are structured to include the following information for each log event (site visit):

- Requestor IP Address
- RFC 1413 Identity
- Username
- Timestamp
- Request method with requested resource and protocol
- HTTP response code to request
- Bytes transferred in request

An example of this might look like the following:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0"
404 2326
```

You could specify a rule which attempts to match events of that structure for HTTP 404 errors, as shown in the following example:

To create a metric filter using the CloudWatch console

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, select a log group, and then click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, in the **Filter Pattern** field, enter `[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]`.
6. To test your filter pattern, in the **Select Log Data to Test** list, select the log group you want to test the metric filter against, and then click **Test Pattern**.
7. Under **Results**, CloudWatch Logs displays a message showing how many occurrences of the filter pattern were found in the log file.

Note

To see detailed results, click **Show test results**.

8. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** field, enter **HTTP404Errors**.
9. Under **Metric Details**, in the **Metric Namespace** field, enter **YourNamespace**.
10. In the **Metric Name** field, enter **ApacheNotFoundCount**, and then click **Create Filter**.

To create a metric filter using the AWS CLI

- At a command prompt, type:

```
% aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name HTTP404Errors \
--filter-pattern '[ip, id, user, timestamp, request, status_code=404,
size]' \
--metric-transformations \
metricName=PageNotFoundCount,metricNamespace=YourNamespace,metricValue=1
```

In this example, literal characters such as the left and right square brackets, double quotes and character string 404 were used. The pattern needs to match with the entire log event message for the log event to be considered for monitoring.

You can verify the creation of the metric filter by using the **describe-monitoring-policies** command. You should see output that looks like this:

```
% aws logs describe-metric-filters --log-group-name MyApp/access.log

{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
                {
                    "metricValue": "1",
                    "metricNamespace": "YourNamespace",
                    "metricName": "PageNotFoundCount"
                }
            ],
            "creationTime": 1399277571078,
            "filterPattern": "[ip, id, user, timestamp, request, status_code=404, size]"
        }
    ]
}
```

Now you can post a few events manually:

```
% aws logs put-log-events \
    --log-group-name MyApp/access.log --log-stream-name hostname \
    --log-events \
    timestamp=1394793518000,message="127.0.0.1 - bob
[10/Oct/2000:13:55:36 -0700] \"GET /apache_pb.gif HTTP/1.0\" 404 2326" \
    timestamp=1394793528000,message="127.0.0.1 - bob
[10/Oct/2000:13:55:36 -0700] \"GET /apache_pb2.gif HTTP/1.0\" 200 2326"
```

Soon after putting these sample log events, you can retrieve the metric named in the CloudWatch console as ApacheNotFoundErrorCode.

Example: Find and Count 400-level into 4xx Metric

As in the previous example, you might want to monitor your web service access logs and monitor the HTTP response code levels. For example, you might want to monitor all of the HTTP 400-level errors. However, you might not want to specify a new metric filter for every return code.

The following example demonstrates how to create a metric that includes all 400-level HTTP code responses from an access log using the Apache access log format from the previous example. The extraction rule matches any characters before "(double quote), matches "(double quote), matches any characters before "(double quote), matches (space), matches numbers, and matches remaining characters.

To create a metric filter using the CloudWatch console

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, select a log group, and then click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, in the **Filter Pattern** field, enter `[ip, id, user, timestamp, request, status_code=4*, size]`.
6. To test your filter pattern, in the **Select Log Data to Test** list, select the log group you want to test the metric filter against, and then click **Test Pattern**.
7. Under **Results**, CloudWatch Logs displays a message showing how many occurrences of the filter pattern were found in the log file.

Note

To see detailed results, click **Show test results**.

8. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** field, enter **HTTP4xxErrors**.
9. Under **Metric Details**, in the **Metric Namespace** field, enter **YourNameSpace**.
10. In the **Metric Name** field, enter **HTTP4xxErrors**, and then click **Create Filter**.

To create a metric filter using the AWS CLI

- At a command prompt, type:

```
% aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name HTTP4xxErrors \
--filter-pattern '[ip, id, user, timestamp, request, status_code=4*, size]' \
--metric-transformations \
metricName=HTTP4xxErrors,metricNamespace=YourNamespace,metricValue=1
```

You can use the following data in put-event calls to test this rule. If you did not remove the monitoring rule in the previous example, you will generate two different metrics.

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200
3
```

Example: Extract Bytes Transferred from an Apache Log

Sometimes, instead of counting, it is helpful to use values within individual log events for metric values. This example shows how you can create an extraction rule to create a metric that measures the bytes transferred by an Apache webserver.

This extraction rule matches the seven fields of the log event. The metric value is the value of the seventh matched token. You can see the reference to the token as "\$7" in the `metricValue` field of the extraction rule.

To create a metric filter using the CloudWatch console

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, select a log group, and then click **Create Metric Filter**.
5. On the **Define Logs Metric Filter** screen, in the **Filter Pattern** field, enter `[ip, id, user, timestamp, request, status_code, size]`.
6. To test your filter pattern, in the **Select Log Data to Test** list, select the log group you want to test the metric filter against, and then click **Test Pattern**.
7. Under **Results**, CloudWatch Logs displays a message showing how many occurrences of the filter pattern were found in the log file.

Note

To see detailed results, click **Show test results**.

8. Click **Assign Metric**, and then on the **Create Metric Filter and Assign a Metric** screen, in the **Filter Name** field, enter `size`.
9. Under **Metric Details**, in the **Metric Namespace** field, enter `YourNameSpace`.
10. In the **Metric Name** field, enter `BytesTransferred`.
11. In the **Metric Value** field, enter `$size`, and then click **Create Filter**.

Note

If the **Metric Value** field isn't visible, click **Show advanced metric settings**.

To create a metric filter using the AWS CLI

- At a command prompt, type:

```
% aws logs put-metric-filter \
--log-group-name MyApp/access.log \
--filter-name BytesTransferred \
--filter-pattern '[ip, id, user, timestamp, request, status_code=4*, size]' \
--metric-transformations \
metricName=BytesTransferred,metricNamespace=YourNamespace,metric
Value=$size
```

You can use the following data in put-log-event calls to test this rule. This generates two different metrics if you did not remove monitoring rule in the previous example.

```
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:49:52 -0700] "GET /index.html HTTP/1.1" 404 287
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /~test/ HTTP/1.1" 200 3
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:50:51 -0700] "GET /favicon.ico HTTP/1.1" 404 308
127.0.0.1 - - [24/Sep/2013:11:51:34 -0700] "GET /~test/index.html HTTP/1.1" 200
3
```

Listing Metric Filters

You can list all metric filters in a log group.

To list metric filters using the CloudWatch console

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, in the list of log groups, in the **Metric Filters** column, click the number of filters.

The **Log Groups > Filters for** screen lists all metric filters associated with the log group.

To list metric filters using the AWS CLI

- At a command prompt, type:

```
% aws logs describe-metric-filters --log-group-name MyApp/access.log

{
    "metricFilters": [
        {
            "filterName": "HTTP404Errors",
            "metricTransformations": [
                {
                    "metricValue": "1",
                    "metricNamespace": "YourNamespace",
                    "metricName": "PageNotFoundCount"
                }
            ],
            "creationTime": 1399277571078,
            "filterPattern": "[ip, id, user, timestamp, request, status_code=404, size]"
        }
    ]
}
```

Deleting a Metric Filter

A policy is identified by its name and the log group it belongs to.

To delete a metric filter using the CloudWatch console

1. Open the Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the region. From the navigation bar, select the region that meets your needs. For more information, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.



3. In the navigation pane, click **Logs**.
4. In the contents pane, in the **Metric Filter** column, click the metric filter you want to delete.
5. On the **Logs Metric Filters** screen, in the metric filter, click **Delete Filter**.
6. In the **Delete Metric filter** dialog box, click **Yes, Delete**.

To delete a metric filter using the AWS CLI

- At a command prompt, type:

```
% aws logs delete-metric-filter --log-group-name MyApp/access.log \
--filter-name YourFilterName
```

Logging Amazon CloudWatch API Calls in AWS CloudTrail

Amazon CloudWatch is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of your AWS account. This information is collected and written to log files that are stored in an Amazon S3 bucket that you specify. API calls are logged when you use the Amazon CloudWatch API, the Amazon CloudWatch console, a back-end console, or the AWS CLI. Using the information collected by CloudTrail, you can determine what request was made to Amazon CloudWatch, the source IP address the request was made from, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#)

Topics

- [CloudWatch Information in CloudTrail \(p. 147\)](#)
- [Understanding Amazon CloudWatch Log File Entries \(p. 148\)](#)

CloudWatch Information in CloudTrail

If CloudTrail logging is turned on, calls made to all Amazon CloudWatch actions are captured in log files. For example, calls to the **EnableAlarmActions**, **PutMetricAlarm**, and **DescribeAlarms** actions generate entries in CloudTrail log files. The following CloudWatch actions are supported:

- PutMetricAlarm
- DescribeAlarms
- DescribeAlarmHistory
- DescribeAlarmsForMetric
- DisableAlarmActions
- EnableAlarmActions
- SetAlarmState
- DeleteAlarms

For more information about these actions, see the [Amazon CloudWatch API Reference](#).

Every log entry contains information about who generated the request. For example, if a request is made to create or update an alarm (**PutMetricAlarm**), CloudTrail logs the user identity of the person or service that made the request. The user identity information helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information about CloudTrail fields, see [CloudTrail Event Reference](#) in the AWS CloudTrail User Guide.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

Understanding Amazon CloudWatch Log File Entries

CloudTrail log files can contain one or more log entries composed of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any input parameters, the date and time of the action, and so on. The log entries do not appear in any particular order. That is, they do not represent an ordered stack trace of the public API calls.

The following log file record shows a user called 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": "GreaterThanOrEqualToThreshold",
                "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 record shows a user called the **SetAlarmState** 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": "SetAlarmState",  
      "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": {  
        "stateValue": "OK",  
        "stateReason": "Test",  
        "alarmName": "CloudTrail Test Alarm"  
      },  
      "responseElements": null,  
      "requestID": "297a5d55-b2d5-11e3-a63d-9b463e6d0ff0",  
      "eventID": "dfaf642c-87d9-418f-9110-1297aa97d41f"  
    },  
    ...additional entries  
  ]  
}
```

The following log file record shows a user called the **EnableAlarmActions** 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": "EnableAlarmActions",  
      "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": {
            "alarmNames": [ "CloudTrail Test Alarm" ]
        },
        "responseElements": null,
        "requestID": "29a516d7-b2d5-11e3-a63d-9b463e6d0ff0",
        "eventID": "acac5afb-8d5c-4a5d-9096-f9f985460969"
    },
    ...additional entries
]
}
```

The following log file record shows a user called the **DisableAlarmActions** 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": "2014-03-23T21:50:35Z",  
      "eventSource": "monitoring.amazonaws.com",  
      "eventName": "DisableAlarmActions",  
      "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": {  
        "alarmNames": ["CloudTrail Test Alarm"]  
      },  
      "responseElements": null,  
      "requestID": "29c34d39-b2d5-11e3-a63d-9b463e6d0ff0",  
      "eventID": "593480d0-44f9-49cc-94c9-7a4757f2e545"  
    },  
    ...additional entries  
  ]  
}
```

The following log file record shows a user called the **DescribeAlarmsForMetric** 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": "2014-03-23T21:50:35Z",  
      "region": "us-east-1",  
      "invocationType": "Event",  
      "awsRegion": "us-east-1",  
      "eventName": "AWS_IAM_API_CALL",  
      "awsPartition": "aws",  
      "sourceIp": "123.45.67.89",  
      "recipientArn": "arn:aws:iam::123456789012:root",  
      "awsSdkVersion": "aws-sdk-go/3.14.0",  
      "awsEventSource": "aws_lambda_function",  
      "awsEventVersion": "1.0",  
      "awsEventSourceARN": "arn:aws:lambda:us-east-1:123456789012:function:myLambdaFunction",  
      "awsEventSignatureVersion": "1",  
      "awsEventSignature": "Signature=...  
    }  
  ]  
}
```

```
        "eventSource": "monitoring.amazonaws.com",
        "eventName": "DescribeAlarmsForMetric",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "10.61.89.162",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux
Seahorse/0.1.0",
        "requestParameters": {
            "namespace": "AWS/CloudWatch",
            "metricName": "CloudTrail Test Metric"
        },
        "responseElements": null,
        "requestID": "2a412fcd-b2d5-11e3-a63d-9b463e6d0ff0",
        "eventID": "51065fac-ef74-4e69-b326-24da63cbdc2e"
    },
    ...additional entries
]
}
```

The following log file record shows a user called the **DescribeAlarms** 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": "2014-03-14T22:37:54Z",
            "eventSource": "monitoring.amazonaws.com",
            "eventName": "DescribeAlarms",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "requestParameters": {
                "alarmNames": ["alarm-xx-123456789012-test_notification_association_manipulation_20100801-TC_MonitorsCRUD20100801--ROYU"]
            },
            "responseElements": null,
            "requestID": "3c6b8f0a-9a0f-44fd-a2be-95a6d032f16e",
            "eventID": "260e3430-884a-4971-a7cf-da2ad378b70c"
        },
        ...additional entries
    ]
}
```

The following log file record shows a user called the **DeleteAlarms** action.

```
{
    "Records": [
        {
            "eventVersion": "1.01",
            "userIdentity": {
                "type": "Root",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:root",
                "accountId": "123456789012",
                "accessKeyId": "EXAMPLE_KEY_ID"
            }
        }
    ]
}
```

```
    "arn": "arn:aws:iam::123456789012:root",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID"
},
"eventTime": "2014-03-14T22:38:05Z",
"eventSource": "monitoring.amazonaws.com",
"eventName": "DeleteAlarms",
"awsRegion": "us-east-1",
"sourceIPAddress": "127.0.0.1",
"requestParameters": {
    "alarmNames": ["alarm-xx-123456789012-test_notification_association_manipulation_20100801-TC_MonitorsCRUD20100801--ROYU"]
},
"responseElements": null,
"requestID": "2ed028b7-7956-44a7-a7ab-044bb774af24",
"eventID": "352f341b-2b28-4e9e-8d57-961d11ce9946"
},
...additional entries
]
}
```

Monitoring Scripts for Amazon EC2 Instances

The Amazon CloudWatch Monitoring Scripts for Amazon Elastic Compute Cloud (Amazon EC2) Linux- and Windows-based instances demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts for Linux are sample Perl scripts that comprise a fully functional example that reports memory, swap, and disk space utilization metrics for a Linux instance. The scripts for Windows are sample PowerShell scripts that comprise a fully functional example that reports memory, page file, and disk space utilization metrics for a Windows instance. You can download the CloudWatch Monitoring Scripts for Linux and for Windows from the Amazon Web Services (AWS) sample code library and install them on your Linux- or Windows-based instances.

Important

These scripts are examples only. They are provided "as is" and are not supported.

Note

Standard Amazon CloudWatch free tier quantities and usage charges for custom metrics apply to your use of these scripts. For more information, see the [Amazon CloudWatch product page](#).

Topics

- [Amazon CloudWatch Monitoring Scripts for Linux \(p. 153\)](#)
- [Amazon CloudWatch Monitoring Scripts for Windows \(p. 160\)](#)

Amazon CloudWatch Monitoring Scripts for Linux

The Amazon CloudWatch Monitoring Scripts for Linux are sample Perl scripts that demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts comprise a fully functional example that reports memory, swap, and disk space utilization metrics for an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance.

Topics

- [Prerequisites \(p. 154\)](#)
- [Getting Started \(p. 155\)](#)
- [Using the Scripts \(p. 156\)](#)

- [Viewing Your Custom Metrics in the AWS Management Console \(p. 160\)](#)

You can download the [Amazon CloudWatch Monitoring Scripts for Linux](#) from the AWS sample code library. The CloudWatchMonitoringScripts-v1.1.0.zip package contains these files:

- **CloudWatchClient.pm**—Shared Perl module that simplifies calling Amazon CloudWatch from other scripts.
- **mon-put-instance-data.pl**—Collects system metrics on an Amazon EC2 instance (memory, swap, disk space utilization) and sends them to Amazon CloudWatch.
- **mon-get-instance-stats.pl**—Queries Amazon CloudWatch and displays the most recent utilization statistics for the EC2 instance on which this script is executed.
- **awscreds.template**—File template for AWS credentials that stores your access key ID and secret access key.
- **LICENSE.txt**—Text file containing the Apache 2.0 license.
- **NOTICE.txt**—copyright notice.

These monitoring scripts are intended for use with Amazon EC2 instances running Linux operating systems. The scripts have been tested on the following Amazon Machine Images (AMIs) for both 32-bit and 64-bit versions:

- Amazon Linux 2011.09.02
- Red Hat Enterprise Linux 6.3
- Ubuntu Server 12.04
- SUSE Linux Enterprise Server 11

Prerequisites

You must perform additional steps on some versions of Linux.

Amazon Linux AMI

If you are running an Amazon Linux AMI, version 2014.03 or later, you'll need to add some additional Perl modules in order for the monitoring scripts to work. Use the following procedure to configure your server.

- Log on to your Amazon Linux AMI instance and install the following packages:

```
sudo yum install perl-Switch perl-Sys-Syslog perl-LWP-Protocol-https
```

Red Hat Enterprise Linux or SUSE Linux

If you are running Red Hat Enterprise Linux or SUSE Linux Enterprise Server, use the following procedure to configure your server.

1. At a command prompt, type `cd /` to switch to the root directory.
2. Type `perl -MCPAN -e shell`, and then type `yes` at every prompt.
3. At the `cpan[1]>` prompt, type `install Bundle::LWP5_837 LWP` to install the LWP bundle and update to LWP version 6.03. Type `yes` at every prompt.

Ubuntu Server

If you are running Ubuntu Server, use the following procedure to configure your server.

- Log on to your Ubuntu Server instance and install the following packages:

```
sudo apt-get install unzip libwww-perl libcrypt-ssleay-perl
```

```
sudo apt-get install libswitch-perl
```

For information on how to connect to Amazon EC2 Linux instances, see [Connecting to Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

Getting Started

The following steps show you how to download, uncompress, and configure the CloudWatch Monitoring Scripts on an EC2 Linux instance.

To download, install, and configure the script

1. Open a command prompt, move to a folder where you want to store the scripts and enter the following:

```
wget http://ec2-downloads.s3.amazonaws.com/cloudwatch-samples/CloudWatchMonitoringScripts-v1.1.0.zip
unzip CloudWatchMonitoringScripts-v1.1.0.zip
rm CloudWatchMonitoringScripts-v1.1.0.zip
cd aws-scripts-mon
```

2. If you already have an AWS Identity and Access Management (IAM) role associated with your instance, make sure that it has permissions to perform the Amazon CloudWatch PutMetricData operation. Otherwise, you can create a new IAM role with permissions to perform CloudWatch operations and associate that role when you launch a new instance. For more information, see [Controlling User Access to Your AWS Account \(p. 170\)](#).
3. Optional: If you aren't using an IAM role, update the `awscreds.template` file that you downloaded earlier. The content of this file should use the following format:

AWSAccessKeyId=*YourAccessKeyId*

AWSSecretKey=*YourSecretAccessKey*

Note

This step is optional if you have already created a file for credentials. You can use an existing file by specifying its location on the command line when you call the scripts. Alternatively, you can set the environment variable `AWS_CREDENTIAL_FILE` to point to the file with your AWS credentials.

For instructions on how to access your credentials, see [Creating, Modifying, and Viewing User Security Credentials](#) in *Using IAM*.

Using the Scripts

mon-put-instance-data.pl

This script collects memory, swap, and disk space utilization data on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
--mem-util	Collects and sends the MemoryUtilization metrics in percentages. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
--mem-used	Collects and sends the MemoryUsed metrics, reported in megabytes. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
--mem-avail	Collects and sends the MemoryAvailable metrics, reported in megabytes. This option reports memory available for use by applications and the operating system.
--swap-util	Collects and sends SwapUtilization metrics, reported in percentages.
--swap-used	Collects and sends SwapUsed metrics, reported in megabytes.
--disk-path=PATH	Selects the disk on which to report. PATH can specify a mount point or any file located on a mount point for the filesystem that needs to be reported. For selecting multiple disks, specify a --disk-path=PATH for each one of them. To select a disk for the filesystems mounted on / and /home, use the following parameters: --disk-path=/ --disk-path=/home
--disk-space-util	Collects and sends the DiskSpaceUtilization metric for the selected disks. The metric is reported in percentages.
--disk-space-used	Collects and sends the DiskSpaceUsed metric for the selected disks. The metric is reported by default in gigabytes. Due to reserved disk space in Linux operating systems, disk space used and disk space available may not accurately add up to the amount of total disk space.
--disk-space-avail	Collects and sends the DiskSpaceAvailable metric for the selected disks. The metric is reported in gigabytes. Due to reserved disk space in the Linux operating systems, disk space used and disk space available may not accurately add up to the amount of total disk space.
--memory-units=UNITS	Specifies units in which to report memory usage. If not specified, memory is reported in megabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.

Name	Description
--disk-space-units=UNITS	Specifies units in which to report disk space usage. If not specified, disk space is reported in gigabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
--aws-credential-file=PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the --aws-access-key-id and --aws-secret-key parameters.
--aws-access-key-id=VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the --aws-secret-key option. Do not use this option with the --aws-credential-file parameter.
--aws-secret-key=VALUE	Specifies the AWS secret access key to use to sign the request to CloudWatch. Must be used together with the --aws-access-key-id option. Do not use this option with --aws-credential-file parameter.
--aws-iam-role=VALUE	Specifies the IAM role used to provide AWS credentials. The value =VALUE is required. If no credentials are specified, the default IAM role associated with the EC2 instance is applied. Only one IAM role can be used. If no IAM roles are found, or if more than one IAM role is found, the script will return an error. Do not use this option with the --aws-credential-file, --aws-access-key-id, or --aws-secret-key parameters.
--aggregated[=only]	Adds aggregated metrics for instance type, AMI ID, and overall for the region. The value =only is optional; if specified, the script reports only aggregated metrics.
--auto-scaling[=only]	Adds aggregated metrics for the Auto Scaling group. The value =only is optional; if specified, the script reports only Auto Scaling metrics. The IAM policy associated with the IAM account or role using the scripts need to have permissions to call the EC2 action DescribeTags .
--verify	Performs a test run of the script that collects the metrics, prepares a complete HTTP request, but does not actually call CloudWatch to report the data. This option also checks that credentials are provided. When run in verbose mode, this option outputs the metrics that will be sent to CloudWatch.
--from-cron	Use this option when calling the script from cron. When this option is used, all diagnostic output is suppressed, but error messages are sent to the local system log of the user account.
--verbose	Displays detailed information about what the script is doing.
--help	Displays usage information.
--version	Displays the version number of the script.

Examples

The following examples assume that you have already updated the `awscreds.conf` file with valid AWS credentials. If you are not using the `awscreds.conf` file, provide credentials using the `--aws-access-key-id` and `--aws-secret-key` arguments.

To perform a simple test run without posting data to CloudWatch

- Run the following command:

```
./mon-put-instance-data.pl --mem-util --verify --verbose
```

To collect all available memory metrics and send them to CloudWatch

- Run the following command:

```
./mon-put-instance-data.pl --mem-util --mem-used --mem-avail
```

To set a cron schedule for metrics reported to CloudWatch

- Start editing the crontab using the following command:

```
crontab -e
```

- Add the following command to report memory and disk space utilization to CloudWatch every five minutes:

```
*/5 * * * * ~/aws-scripts-mon/mon-put-instance-data.pl --mem-util --disk-space-util --disk-path=/ --from-cron
```

If the script encounters an error, the script will write the error message in the system log.

To collect aggregated metrics for an Auto Scaling group and send them to Amazon CloudWatch without reporting individual instance metrics

- Run the following command:

```
./mon-put-instance-data.pl --mem-util --mem-used --mem-avail --auto-scaling-only
```

To collect aggregated metrics for instance type, AMI ID and region, and send them to Amazon CloudWatch without reporting individual instance metrics

- Run the following command:

```
./mon-put-instance-data.pl --mem-util --mem-used --mem-avail --aggregated=only
```

mon-get-instance-stats.pl

This script queries CloudWatch for statistics on memory, swap, and disk space metrics within the time interval provided using the number of most recent hours. This data is provided for the Amazon EC2 instance on which this script is executed.

Options

Name	Description
--recent-hours=N	Specifies the number of recent hours to report on, as represented by N where N is an integer.
--aws-credential-file=PATH	Provides the location of the file containing AWS credentials.
--aws-access-key-id=VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the --aws-secret-key option. Do not use this option with the --aws-credential-file option.
--aws-secret-key=VALUE	Specifies the AWS secret access key to use to sign the request to CloudWatch. Must be used together with the --aws-access-key-id option. Do not use this option with --aws-credential-file option.
--aws-iam-role=VALUE	Specifies the IAM role used to provide AWS credentials. The value =VALUE is required. If no credentials are specified, the default IAM role associated with the EC2 instance is applied. Only one IAM role can be used. If no IAM roles are found, or if more than one IAM role is found, the script will return an error. Do not use this option with the --aws-credential-file, --aws-access-key-id, or --aws-secret-key parameters.
--verify	Performs a test run of the script that collects the metrics, prepares a complete HTTP request, but does not actually call CloudWatch to report the data. This option also checks that credentials are provided. When run in verbose mode, this option outputs the metrics that will be sent to CloudWatch.
--verbose	Displays detailed information about what the script is doing.
--help	Displays usage information.
--version	Displays the version number of the script.

Examples

To get utilization statistics for the last 12 hours

- Run the following command:

```
mon-get-instance-stats.pl --recent-hours=12
```

The returned response will be similar to the following example output:

```
Instance metric statistics for the last 12 hours.

CPU Utilization
    Average: 1.06%, Minimum: 0.00%, Maximum: 15.22%

Memory Utilization
    Average: 6.84%, Minimum: 6.82%, Maximum: 6.89%

Swap Utilization
    Average: N/A, Minimum: N/A, Maximum: N/A

Disk Space Utilization on /dev/xvda1 mounted as /
    Average: 9.69%, Minimum: 9.69%, Maximum: 9.69%
```

Viewing Your Custom Metrics in the AWS Management Console

If you successfully call the mon-put-instance-data.pl script, you can use the AWS Management Console to view your posted custom metrics in the Amazon CloudWatch console.

To view custom metrics

1. Execute mon-put-instance-data.pl, as described earlier.
2. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
3. Click **View Metrics**.
4. In the **Viewing** drop-down list, your custom metrics posted by the script display with the prefix System/Linux.

Amazon CloudWatch Monitoring Scripts for Windows

The Amazon CloudWatch Monitoring Scripts for Windows are sample PowerShell scripts that demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts comprise a fully functional example that reports memory, page file, and disk space utilization metrics for an Amazon Elastic Compute Cloud (Amazon EC2) Windows instance.

Topics

- [Getting Started \(p. 161\)](#)
- [Using the Scripts \(p. 162\)](#)

You can download [Amazon CloudWatch Monitoring Scripts for Microsoft Windows Server](#) from the Amazon Web Services (AWS) sample code library. The AmazonCloudWatchMonitoringWindows.zip package contains these files:

- **mon-put-metrics-mem.ps1**—Collects system metrics on an Amazon EC2 Windows instance (memory, page file utilization) and sends them to Amazon CloudWatch.

- **mon-put-metrics-disk.ps1**—Collects system metrics on an Amazon EC2 instance (disk space utilization) and sends them to Amazon CloudWatch.
- **mon-put-metrics-perfmon.ps1**—Collects PerfMon counters on an Amazon EC2 instance and sends them to Amazon CloudWatch.
- **mon-get-instance-stats.ps1**—Queries Amazon CloudWatch and displays the most recent utilization statistics for the EC2 instance on which this script is executed.
- **awscreds.conf**—File template for AWS credentials that stores your access key ID and secret access key.
- **LICENSE.txt**—Text file containing the Apache 2.0 license.
- **NOTICE.txt**—Copyright notice.

These monitoring scripts are intended for use with Amazon EC2 instances running Microsoft Windows Server. The scripts have been tested on the following Amazon Machine Images (AMIs) for both 32-bit and 64-bit versions:

- Windows Server 2003 R2
- Windows Server 2008
- Windows Server 2008 R2

Getting Started

The following steps demonstrate how to download, uncompress, and configure the Amazon CloudWatch Monitoring Scripts on an Amazon EC2 Windows instance.

To download, install, and configure the script

1. Connect to your Amazon EC2 Windows instance. For information about how to connect to Amazon EC2 Windows instances, see [Connecting to Windows Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Download and install the [AWS SDK for .NET](#) onto the EC2 instance that you want to monitor.
3. Download the .zip file containing the [Amazon CloudWatch Monitoring Scripts for Microsoft Windows Server](#) onto the EC2 instance and unzip it in a location of your preference.
4. Update the awscreds.conf file that you downloaded earlier. The content of this file should use the following format:

AWSAccessKeyId=*YourAccessKeyId*

AWSSecretKey=*YourSecretAccessKey*

Note

This step is optional if you have already created a file for credentials. You can use an existing file by specifying its location on the command line when you call the scripts. Alternatively, you can set the environment variable `AWS_CREDENTIAL_FILE` to point to the file with your AWS credentials.

For instructions on how to access your credentials, use the following procedure.

As a best practice, do not use the root credentials. Instead you should create an Identity and Access Management (IAM) user with a policy that restricts the user to only Amazon CloudWatch operations. For more information, see [Controlling User Access to Your AWS Account](#) (p. 170)

Using the Scripts

mon-put-metrics-mem.ps1

This script collects memory and pagefile utilization data on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
<code>-mem_util</code>	Collects and sends the MemoryUtilization metrics in percentages. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
<code>-mem_used</code>	Collects and sends the MemoryUsed metrics, reported in megabytes. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
<code>-mem_avail</code>	Collects and sends the MemoryAvailable metrics, reported in megabytes. This option reports memory available for use by applications and the operating system.
<code>-page_util</code>	Collects and sends PageUtilization metrics, reported in percentages. Page utilization is reported for each page file in a windows instance.
<code>-page_used</code>	Collects and sends PageUsed metrics, reported in megabytes.
<code>-page_avail</code>	Reports available space in page file for all disks.
<code>-memory_units UNITS</code>	Specifies units in which to report memory usage. If not specified, memory is reported in megabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
<code>-aws_credential_file=PATH</code>	Provides the location of the file containing AWS credentials. This parameter cannot be used with the <code>-aws_access_id</code> and <code>-aws_secret_key</code> parameters.
<code>-aws_access_id=VALUE</code>	Specifies the AWS access key ID to use to identify the caller. Must be used together with the <code>-aws_secret_key</code> option. Do not use this option with the <code>-aws_credential_file</code> option.
<code>-aws_secret_key=VALUE</code>	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the <code>-aws_access_key_id</code> option. Do not use this option with <code>-aws_credential_file</code> option.
<code>-whatif</code>	Performs a test run of the script that collects the metrics but does not actually call Amazon CloudWatch to report the data. This option also checks that credentials are provided.
<code>-from_scheduler</code>	Use this option when calling the script from task scheduler. When this option is used, all diagnostic output is suppressed, but error messages are sent to the log file.
<code>-verbose</code>	Displays detailed information about what the script is doing.

Name	Description
<i>Get-help mon-put-metrics-mem.ps1</i>	Displays usage information.
<i>-version</i>	Displays the version number of the script.
<i>-logfile</i>	LogFile is used to log error message. Use this along with <i>-from_scheduler</i> option. If no value is specified for logfile then a default file is created with the same as the script with .log extension.

Examples

The following examples assume that you have already updated the `awscreds.conf` file with valid AWS credentials. If you are not using the `awscreds.conf` file, provide credentials using the `-aws_access_id` and `-aws_secret_key` arguments.

To collect all available memory metrics using an inline access ID and secret key and send the data to CloudWatch

- Run the following command:

```
.\mon-put-metrics-mem.ps1 -aws_access_id ThisIsMyAccessKey -aws_secret_key
ThisIsMySecretKey -mem_util -mem_avail -page_avail -page_used -page_util
-memory_units Megabytes
```

To collect all available memory metrics using a credential file and send the data to CloudWatch

- Run the following command:

```
.\mon-put-metrics-mem.ps1 -aws_credential_file C:\awscreds.conf -mem_util
-mem_used -mem_avail -page_avail -page_used -page_util -memory_units Megabytes
```

To collect all available memory metrics using credentials stored in environment variables and send the data to CloudWatch

- Run the following command:

```
.\mon-put-metrics-mem.ps1 -mem_util -mem_used -mem_avail -page_avail -
page_used -page_util -memory_units Megabytes
```

mon-put-metrics-disk.ps1

This script collects disk space utilization data on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
<code>-disk_space_util</code>	Collects and sends the DiskSpaceUtilization metric for the selected disks. The metric is reported in percentages.
<code>-disk_space_used</code>	Collects and sends DiskSpaceUsed metric for the selected disks. The metric is reported by default in gigabytes.
<code>-disk_space_avail</code>	Collects and sends the DiskSpaceAvailable metric for the selected disks. The metric is reported in gigabytes.
<code>-disk_space_units UNITS</code>	Specifies units in which to report memory usage. If not specified, memory is reported in gigabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
<code>-disk_drive</code>	Selects the drive letter on which to report. To report metrics on the c and d drives, use the following option <code>-disk_drive C:, D:</code> Values should be comma separated.
<code>-aws_credential_file PATH</code>	Provides the location of the file containing AWS credentials. This parameter cannot be used with the <code>-aws_access_id</code> and <code>-aws_secret_key</code> parameters.
<code>-aws_access_id VALUE</code>	Specifies the AWS access key ID to use to identify the caller. Must be used together with the <code>-aws_secret_key</code> option. Do not use this option with the <code>-aws_credential_file</code> option.
<code>-aws_secret_key VALUE</code>	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the <code>-aws_access_id</code> option. Do not use this option with <code>-aws_credential_file</code> option.
<code>-whatif</code>	Performs a test run of the script that collects the metrics but does not actually call Amazon CloudWatch to report the data. This option also checks that credentials are provided.
<code>-from_scheduler</code>	Use this option when calling the script from task scheduler. When this option is used, all diagnostic output is suppressed, but error messages are sent to the log file.
<code>-verbose</code>	Displays detailed information about what the script is doing.
<code>Get-help mon-put-metrics-disk.ps1</code>	Displays usage information.
<code>-version</code>	Displays the version number of the script.
<code>-logfile</code>	Logfile is used to log error message. Use this along with <code>-from_scheduler</code> option. If no value is specified for logfile then a default file is created with the same as the script with .log extension.

Examples

To collect all available disk metrics using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-disk.ps1 -aws_access_id ThisIsMyAccessKey -aws_secret_key  
ThisIsMySecretKey -disk_space_util -disk_space_avail -disk_space_units  
Gigabytes
```

To collect all available disk metrics using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-disk.ps1  
-aws_credential_file C:\awscreds.conf -disk_drive C:, D:  
-disk_space_util -disk_space_used -disk_space_avail -disk_space_units  
Gigabytes
```

To collect all available disk metrics using credentials stored in an environment variable and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-disk.ps1 -disk_drive C:, D:  
-disk_space_util -disk_space_used -disk_space_avail -  
disk_space_units Gigabytes
```

mon-put-metrics-perfmon.ps1

This script collects PerfMon counters on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
-processor_queue	Reports current processor queue counter.
-pages_input	Reports memory pages/input memory counter.
-aws_credential_file PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the -aws_access_id and -aws_secret_key parameters.

Name	Description
<code>-aws_access_id VALUE</code>	Specifies the AWS access key ID to use to identify the caller. Must be used together with the <code>-aws_secret_key</code> option. Do not use this option with the <code>-aws_credential_file</code> option.
<code>-aws_secret_key VALUE</code>	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the <code>-aws_access_id</code> option. Do not use this option with <code>-aws_credential_file</code> option.
<code>-whatif</code>	Performs a test run of the script that collects the metrics but does not actually call Amazon CloudWatch to report the data. This option also checks that credentials are provided.
<code>-from_scheduler</code>	Use this option when calling the script from task scheduler. When this option is used, all diagnostic output is suppressed, but error messages are sent to the log file.
<code>-verbose</code>	Displays detailed information about what the script is doing.
<code>Get-help mon-put-metrics-disk.ps1</code>	Displays usage information.
<code>-version</code>	Displays the version number of the script.
<code>-logfile</code>	LogFile is used to log error message. Use this along with <code>-from_scheduler</code> option. If no value is specified for logfile then a default file is created with the same as the script with .log extension.

Examples

To collect preset PerfMon counters in script using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-perfmon.ps1 -aws_access_id ThisIsMyAccessKey -aws_secret_key
ThisIsMySecretKey -pages_input -processor_queue
```

To collect preset PerfMon counters in script using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-perfmon.ps1 -aws_credential_file C:\awscreds.conf -
pages_input -processor_queue
```

To collect preset PerfMon counters in script using credentials stored in an environment variable and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-perfmon.ps1 -pages_input -processor_queue
```

To add more counters to be pushed to Amazon CloudWatch

- Open the script in a text editor such as Notepad, and then on line 72, locate the following commented section:

```
### Add More counters here.  
#$Counters.Add('\Memory\Cache Bytes','Bytes')  
#$Counters.Add('\\localhost\physicaldisk(0 c:)\% disk time','Percent')
```

Note

The first parameter (e.g., \$Counters.Add) is the PerfMon counter. The second parameter (e.g., ('\Memory\Cache Bytes','Bytes')) is the unit of data that counter provides.

- Edit the script and add your own PerfMon counters to the script as shown above. After you have added custom PerfMon counters to the script, you can run the script without any parameters other than credential information.

Note

You can only add PerfMon counters to the script on your computer. You can use the Get-Counter command to test PerfMon counters. For more information, see [Get-Counter](#) on the Microsoft TechNet website.

mon-get-instance-stats.ps1

This script queries Amazon CloudWatch for statistics on memory, page file, and disk space metrics within the time interval provided using the number of most recent hours. This data is provided for the Amazon EC2 instance on which this script is executed.

Options

Name	Description
<code>-recent-hours N</code>	Specifies the number of recent hours to report on, as represented by N where N is an integer.
<code>-aws_credential_file PATH</code>	Provides the location of the file containing AWS credentials. This parameter cannot be used with the <code>-aws_access_id</code> and <code>-aws_secret_key</code> parameters.
<code>-aws_access_id VALUE</code>	Specifies the AWS access key ID to use to identify the caller. Must be used together with the <code>-aws_secret_key</code> option. Do not use this option with the <code>-aws_credential_file</code> option.
<code>-aws_secret_key VALUE</code>	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the <code>-aws_access_id</code> option. Do not use this option with <code>-aws_credential_file</code> option.

Name	Description
<code>-verbose</code>	Displays detailed information about what the script is doing.
<code>Get-help mon-get-instance-stats.ps1</code>	Displays usage information.
<code>-version</code>	Displays the version number of the script.

Examples

To get utilization statistics for the last 12 hours using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\ mon-get-instance-stats.ps1 -aws_access_id
                               ThisIsMyAccessKey -aws_secret_key ThisIsMySecretKey -recent_hours 12
```

To get utilization statistics for the last 12 hours using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-get-instance-stats.ps1 -aws_credential_file C:\awscreds.conf -recent_hours 12
```

To get utilization statistics for the last 12 hours using credentials stored in an environment variable and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-get-instance-stats.ps1 -recent_hours 12
```

The returned response will be similar to the following example output:

```
Assembly Loaded
Instance Metrics for last 12 hours.
CPU Utilization
Average: 4.69 % Maximum: 10.47 % Minimum: 1.16 %

Memory Utilization
Average: 14.45 % Maximum: 14.77 % Minimum: 14.38 %

pagefileUtilization(c:\pagefile.sys)
Average: 0.00 % Maximum: 0.00 % Minimum: 0.00 %
```

```
Volume Utilization C:  
Average: 17.28 % Maximum: 17.28 % Minimum: 17.28 %  
  
Volume Utilization D:  
Average: 1.41 % Maximum: 1.41 % Minimum: 1.41 %  
  
pagefileUtilization(f:\pagefile.sys)  
Average: 0.00 % Maximum: 0.00 % Minimum: 0.00 %  
pagefileUtilization(f:\pagefile.sys)  
Average: 0 Maximum: 0 Minimum: 0  
  
pagefileUtilization(f:\pagefile.sys)  
Average: 0 Maximum: 0 Minimum: 0
```

Set Up Task Scheduler to Send Metrics Reports to Amazon CloudWatch

You can use Windows Task Scheduler to send metrics reports periodically to Amazon CloudWatch.

To set up task scheduler to send metrics reports to Amazon CloudWatch

1. On your Windows Server instance, click **Start**, click **Administrative Tools**, and then click **Task Scheduler**.
2. On the **Action** menu, click **Create Task**.
3. In the **Create Task** dialog box, on the **General** tab, in the **Name** box, type a name for the task, and then select **Run whether user is logged on or not**.
4. On the **Triggers** tab, click **New**.
5. In the **New Trigger** dialog box, under **Settings**, select **One time**.
6. Under **Advanced settings**, select **Repeat task every** and select **5 minutes** from the drop-down menu.
7. In the **for a duration of** drop-down menu, select **Indefinitely**, and then click **OK**.

Note

These settings create a trigger that will launch the script every 5 minutes indefinitely. To modify this task to run for set number of days using the **Expire** check box.

8. On the **Actions** tab, click **New**.
9. In the **Action** drop-down menu, select **Start a program**.
10. Under **Settings**, in the **Program/script** box, type **Powershell.exe**.
11. In the **Add arguments (optional)** box, type **-command "C:\scripts\mon-put-metrics-disk.ps1 -disk_drive C:,d -disk_space_util -disk_space_units gigabytes -from_scheduler -logfile C:\mylogfile.log"**, and then click **OK**.
12. On the **Create Task** dialog box, click **OK**.

If you selected a user account to run this task, Task Scheduler will prompt you for user credentials. Enter the user name and password for the account that will run the task, and then click **OK**.

Note

If the PerfMon counters you are using don't require administrator privileges, you can run this task using a system account instead of an administrator account. In the **Create Task** dialog box, on the **General** tab, click **Change User or Group**, and then select a system account.

Controlling User Access to Your AWS Account

Amazon CloudWatch integrates with AWS Identity and Access Management (IAM) so that you can specify which CloudWatch actions a user in your AWS Account can perform. For example, you could create an IAM policy that gives only certain users in your organization permission to use `GetMetricStatistics`. They could then use the action to retrieve data about your cloud resources.

You can't use IAM to control access to CloudWatch data for specific resources. For example, you can't give a user access to CloudWatch data for only a specific set of instances or a specific LoadBalancer. Permissions granted using IAM cover all the cloud resources you use with CloudWatch. In addition, you can't use IAM roles with the Amazon CloudWatch command line tools.

Important

Using Amazon CloudWatch with IAM doesn't change how you use CloudWatch. There are no changes to CloudWatch actions, and no new CloudWatch actions related to users and access control.

For an example of a policy that covers CloudWatch actions, see [Example Policies for CloudWatch \(p. 172\)](#).

Topics

- [Amazon CloudWatch ARNs \(p. 170\)](#)
- [CloudWatch Actions \(p. 171\)](#)
- [CloudWatch Keys \(p. 171\)](#)
- [Example Policies for CloudWatch \(p. 172\)](#)

Amazon CloudWatch ARNs

CloudWatch doesn't have any specific resources for you to control access to. Therefore, there are no CloudWatch ARNs for you to use in an IAM policy. You use * as the resource when writing a policy to control access to CloudWatch actions. For more information about ARNs, go to [ARNs in Using IAM](#).

However, if you are using either the Amazon CloudWatch CLI or API, or if you are using the AWS SDKs with the API, to create an Amazon CloudWatch alarm using an Amazon EC2 instance metric, you can add an action using the action's dedicated Amazon Resource Name (ARN). You can add the action to any alarm state, and you can specify the region for each action. The region must match the region to

which you send the put-metric-alarm request. For more information, see [Using the CLI or the API to Create an Alarm to Stop or Terminate an Instance \(p. 96\)](#).

Action	ARN (with region)
<i>Stop</i>	arn:aws:automate:us-east-1:ec2:stop
<i>Terminate</i>	arn:aws:automate:us-east-1:ec2:terminate

CloudWatch Actions

In an IAM policy, you can specify any and all actions that CloudWatch offers. The action name must be prefixed with the lowercase string `cloudwatch:`, `ec2:`, or `autoscaling:`. For example: `cloudwatch:GetMetricStatistics`, `cloudwatch>ListMetrics`, or `cloudwatch:*` (for all CloudWatch actions). The actions you can specify in an IAM policy for use with CloudWatch are listed below.

Service	Action
Amazon CloudWatch	<code>cloudwatch>DeleteAlarms</code>
Amazon CloudWatch	<code>cloudwatch>DescribeAlarmHistory</code>
Amazon CloudWatch	<code>cloudwatch>DescribeAlarms</code>
Amazon CloudWatch	<code>cloudwatch>DescribeAlarmsForMetric</code>
Amazon CloudWatch	<code>cloudwatch>DisableAlarmActions</code>
Amazon CloudWatch	<code>cloudwatch>EnableAlarmActions</code>
Amazon CloudWatch	<code>cloudwatch>GetMetricStatistics</code>
Amazon CloudWatch	<code>cloudwatch>ListMetrics</code>
Amazon CloudWatch	<code>cloudwatch>PutMetricAlarm</code>
Amazon CloudWatch	<code>cloudwatch>PutMetricData</code>
Amazon CloudWatch	<code>cloudwatch>SetAlarmState</code>
Amazon EC2	<code>ec2>DescribeInstanceStatus</code>
Amazon EC2	<code>ec2>DescribeInstances</code>
Amazon EC2	<code>ec2>StopInstances</code>
Amazon EC2	<code>ec2>TerminateInstances</code>
Auto Scaling	<code>autoscaling>Scaling</code>
Auto Scaling	<code>autoscaling>Trigger</code>

CloudWatch Keys

CloudWatch implements the following policy keys, but no others. For more information about policy keys, go to [Condition](#) in [Using IAM](#).

AWS-Wide Policy Keys

- `aws:CurrentTime`—To check for date/time conditions.
- `aws:EpochTime`—To check for date/time conditions using a date in epoch or UNIX time.
- `aws:MultiFactorAuthAge`—To check how long ago (in seconds) the MFA-validated security credentials making the request were issued using Multi-Factor Authentication (MFA). Unlike other keys, if MFA is not used, this key is not present.
- `aws:principalType`—To check the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SecureTransport`—To check whether the request was sent using SSL. For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.
- `aws:SourceArn`—To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see [Amazon Resource Name \(ARN\)](#) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`—To check the IP address of the requester. Note that if you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, the public IP address of the instance is evaluated.
- `aws:UserAgent`—To check the client application that made the request.
- `aws:userid`—To check the user ID of the requester.
- `aws:username`—To check the user name of the requester, if available.

Note

Key names are case sensitive.

Example Policies for CloudWatch

Some sample policies for controlling user access to Amazon CloudWatch are shown below. For more information about the sections within an IAM policy statement, see [IAM Policy Elements Reference](#) in *Using IAM*.

Note

In the future, CloudWatch might add new actions that should logically be included in the following policy, based on the policy's stated goals.

Example Policy for Retrieving CloudWatch Data

The following sample policy allows a group to retrieve CloudWatch data, but only if the group uses SSL with the request.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["cloudwatch:GetMetricStatistics", "cloudwatch>ListMetrics"],  
            "Resource": "*",  
            "Condition": {  
                "Bool": {  
                    "aws:SecureTransport": "true"  
                }  
            }  
        }  
    ]  
}
```

Example Policy for Stopping or Terminating an Amazon EC2 Instance

The following sample policy allows an CloudWatch alarm action to stop or terminate an Amazon 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": "0000000000000000",  
            "Resource": [  
                "*"  
            ],  
            "Effect": "Allow"  
        },  
        {  
            "Action": [  
                "ec2:DescribeInstanceStatus",  
                "ec2:DescribeInstances",  
                "ec2:StopInstances",  
                "ec2:TerminateInstances"  
            ],  
            "Sid": "0000000000000000",  
            "Resource": [  
                "*"  
            ],  
            "Effect": "Allow"  
        }  
    ]  
}
```

For more information about IAM, see:

- [AWS Identity and Access Management \(IAM\)](#)
- [IAM Getting Started Guide](#)
- [Using IAM](#)

Amazon CloudWatch Namespaces, Dimensions, and Metrics Reference

This section includes all of the namespaces, dimensions, and metrics that you can use with CloudWatch. Namespaces are containers for metrics. Metrics, which are time-ordered sets of data points, are isolated from one another in different namespaces so that metrics from different applications are not mistakenly aggregated into the same statistics. In addition, each metric has a dimension, which is a name/value pair that helps you to uniquely identify a metric.

Topics

- [AWS Namespaces \(p. 176\)](#)
- [Auto Scaling Dimensions and Metrics \(p. 176\)](#)
- [AWS Billing Dimensions and Metrics \(p. 179\)](#)
- [Amazon CloudFront Dimensions and Metrics \(p. 180\)](#)
- [DynamoDB Dimensions and Metrics \(p. 181\)](#)
- [Amazon ElastiCache Dimensions and Metrics \(p. 186\)](#)
- [Amazon EBS Dimensions and Metrics \(p. 191\)](#)
- [Amazon Elastic Compute Cloud Dimensions and Metrics \(p. 192\)](#)
- [Elastic Load Balancing Dimensions and Metrics \(p. 196\)](#)
- [Amazon Elastic MapReduce Dimensions and Metrics \(p. 198\)](#)
- [Amazon Kinesis Dimensions and Metrics \(p. 202\)](#)
- [AWS OpsWorks Dimensions and Metrics \(p. 203\)](#)
- [Amazon Redshift Dimensions and Metrics \(p. 205\)](#)
- [Amazon RDS Dimensions and Metrics \(p. 208\)](#)
- [Amazon Route 53 Dimensions and Metrics \(p. 210\)](#)
- [Amazon Simple Notification Service Dimensions and Metrics \(p. 210\)](#)
- [Amazon SQS Dimensions and Metrics \(p. 211\)](#)
- [Amazon SWF Dimensions and Metrics \(p. 213\)](#)
- [AWS Storage Gateway Dimensions and Metrics \(p. 214\)](#)

AWS Namespaces

Namespaces are containers for metrics. All AWS services that provide Amazon CloudWatch data use a namespace string, beginning with "AWS/". The following services push metric data points to CloudWatch.

AWS Product	Namespace
Auto Scaling	AWS/AutoScaling
AWS Billing	AWS/Billing
Amazon CloudFront	AWS/CloudFront
Amazon DynamoDB	AWS/DynamoDB
Amazon ElastiCache	AWS/ElastiCache
Amazon Elastic Block Store	AWS/EBS
Amazon Elastic Compute Cloud	AWS/EC2
Elastic Load Balancing	AWS/ELB
Amazon Elastic MapReduce	AWS/ElasticMapReduce
Amazon Kinesis	AWS/Kinesis
AWS OpsWorks	AWS/OpsWorks
Amazon Redshift	AWS/Redshift
Amazon Relational Database Service	AWS/RDS
Amazon Route 53	AWS/Route53
Amazon Simple Notification Service	AWS/SNS
Amazon Simple Queue Service	AWS/SQS
Amazon Simple Workflow Service	AWS/SWF
AWS Storage Gateway	AWS/StorageGateway

Auto Scaling Dimensions and Metrics

This section discusses the metrics that Auto Scaling instances and groups send to CloudWatch and describes how to enable detailed (one-minute) monitoring and basic (five-minute) monitoring. For more information about how to monitor Auto Scaling, see [Monitor Your Auto Scaling Instances in the Auto Scaling Developer Guide](#).

Topics

- [Auto Scaling Instance Support \(p. 177\)](#)
- [Auto Scaling Group Support \(p. 178\)](#)

Auto Scaling Instance Support

This section discusses the metrics that Auto Scaling instances send to CloudWatch. Instance metrics are the metrics that an individual Amazon EC2 instance sends to CloudWatch. Instance metrics are the same metrics available for any Amazon EC2 instance, whether or not it is in an Auto Scaling group. For information about instance metrics for Amazon EC2 instances, see [Amazon Elastic Compute Cloud Dimensions and Metrics \(p. 192\)](#).

CloudWatch offers basic or detailed monitoring. Basic monitoring sends aggregated data about each instance to CloudWatch every five minutes. Detailed monitoring offers more frequent aggregated data by sending data from each instance every minute.

Note

Selecting detailed monitoring is a prerequisite for the collection of Auto Scaling group metrics. For more information, see [the section called “Auto Scaling Group Support” \(p. 178\)](#).

The following sections describe how to enable either detailed monitoring or basic monitoring.

Activating Detailed Instance Monitoring for Auto Scaling

To enable detailed instance monitoring for a new Auto Scaling group, you don't need to take any extra steps. One of your first steps when creating an Auto Scaling group is to create a launch configuration. Each launch configuration contains a flag named `InstanceMonitoring.Enabled`. The default value of this flag is `true`, so you don't need to set this flag if you want detailed monitoring.

If you have an Auto Scaling group for which you have explicitly selected basic monitoring, the switch to detailed monitoring involves several steps, especially if you have CloudWatch alarms configured to scale the group automatically.

To switch to detailed instance monitoring for an existing Auto Scaling group

1. Create a launch configuration that has the `InstanceMonitoring.Enabled` flag enabled. If you are using the command line tools, create a launch configuration with the `--monitoring-enabled` option.
2. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will enable detailed monitoring for new instances that it creates.
3. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

To...	Do This...
Preserve existing instances	Call <code>MonitorInstances</code> from the Amazon EC2 API for each existing instance to enable detailed monitoring.
Terminate existing instances	Call <code>TerminateInstanceInAutoScalingGroup</code> from the Auto Scaling API for each existing instance. Auto Scaling will use the updated launch configuration to create replacement instances with detailed monitoring enabled.

4. If you have CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the CloudWatch API to update each alarm so that the alarm's period value is set to 60 seconds.

Activating Basic Instance Monitoring for Auto Scaling

To create a new Auto Scaling group with basic monitoring instead of detailed monitoring, associate your new Auto Scaling group with a launch configuration that has the `InstanceMonitoring.Enabled` flag set to `false`. If you are using the command line tools, create a launch configuration with the `--monitoring-disabled` option.

To switch to basic instance monitoring for an existing Auto Scaling group

1. Create a launch configuration that has the `InstanceMonitoring.Enabled` flag disabled. If you are using the command line tools, create a launch configuration with the `--monitoring-disabled` option.
2. If you previously enabled group metrics with a call to `EnableMetricsCollection`, call `DisableMetricsCollection` on your Auto Scaling group to disable collection of all group metrics. For more information, see [the section called “Auto Scaling Group Support” \(p. 178\)](#).
3. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will disable detailed monitoring for new instances that it creates.
4. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

To...	Do This...
Preserve existing instances	Call <code>UnmonitorInstances</code> from the Amazon EC2 API for each existing instance to disable detailed monitoring.
Terminate existing instances	Call <code>TerminateInstanceInAutoScalingGroup</code> from the Auto Scaling API for each existing instance. Auto Scaling will use the updated launch configuration to create replacement instances with detailed monitoring disabled.

5. If you have CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the CloudWatch API to update each alarm so that the alarm's period value is set to 300 seconds.

Important

If you do not update your alarms to match the five-minute data aggregations, your alarms will continue to check for statistics every minute and might find no data available for as many as four out of every five periods.

For more information about instance metrics for Amazon EC2 instances, see [the section called “Amazon Elastic Compute Cloud Dimensions and Metrics” \(p. 192\)](#).

Auto Scaling Group Support

Group metrics are metrics that an Auto Scaling group sends to CloudWatch to describe the group rather than any of its instances. If you enable group metrics, Auto Scaling sends aggregated data to CloudWatch every minute. If you disable group metrics, Auto Scaling does not send any group metrics data to CloudWatch.

To enable group metrics

1. Enable detailed instance monitoring for the Auto Scaling group by setting the `InstanceMonitoring.Enabled` flag in the Auto Scaling group's launch configuration. For more information, see [Auto Scaling Instance Support \(p. 177\)](#).

2. Call `EnableMetricsCollection`, which is part of the Auto Scaling Query API. Alternatively, you can use the equivalent `as-enable-metrics-collection` command that is part of the Auto Scaling command line tools.

Auto Scaling group metrics table

You may enable or disable each of the following metrics, separately.

Metric	Description
GroupMinSize	The minimum size of the Auto Scaling group.
GroupMaxSize	The maximum size of the Auto Scaling group.
GroupDesiredCapacity	The number of instances that the Auto Scaling group attempts to maintain.
GroupInServiceInstances	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating.
GroupPendingInstances	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating.
GroupStandbyInstances	The number of instances that are in a <code>Standby</code> state. Instances in this state are still running but are not actively in service. This metric is not included by default; you must request it specifically.
GroupTerminatingInstances	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending.
GroupTotalInstances	The total number of instances in the Auto Scaling group. This metric identifies the number of instances that are in service, pending, and terminating.

Dimensions for Auto Scaling Group Metrics

The only dimension that Auto Scaling sends to CloudWatch is the name of the Auto Scaling group. This means that all available statistics are filtered by Auto Scaling group name.

AWS Billing Dimensions and Metrics

AWS Billing Metrics

Metric	Description
EstimatedCharges	The estimated charges for your AWS usage. This can either be estimated charges for one service or a roll-up of estimated charges for all services.

Dimensions for AWS Billing Metrics

AWS Billing sends the ServiceName and LinkedAccount dimensions to CloudWatch.

Dimension	Description
ServiceName	The name of the AWS service. This dimension is omitted for the total of estimated charges across all services.
LinkedAccount	The linked account number. This is used for consolidated billing only. This dimension is included only for accounts that are linked to a separate paying account in a consolidated billing relationship. It is not included for accounts that are not linked to a consolidated billing paying account.
Currency	The monetary currency to bill the account. This dimension is required. Unit: USD

Amazon CloudFront Dimensions and Metrics

The metrics and dimensions that CloudFront sends to CloudWatch are listed below. For more information about how to monitor CloudFront, see [Monitoring CloudFront Activity Using CloudWatch](#) in the *Amazon CloudFront Developer Guide*.

Note

Metrics for CloudFront are available only in the US East (N. Virginia) region.

Amazon CloudFront Metrics

Note

Only one statistic, Average or Sum, is applicable for each metric. However, all statistics are available through the console, API, and AWS Command Line Interface. In the following table, each metric specifies the statistic that is applicable to that metric.

Metric	Description
Requests	The number of requests for all HTTP methods and for both HTTP and HTTPS requests. Valid Statistic: Sum Units: Count
BytesDownloaded	The number of bytes downloaded by viewers for GET, HEAD, and OPTIONS requests. Valid Statistic: Sum Units: Bytes

Metric	Description
BytesUploaded	The number of bytes uploaded to your origin with CloudFront using POST and PUT requests. Valid Statistic: Sum Units: Bytes
TotalErrorRate	The percentage of all requests for which the HTTP status code is 4xx or 5xx. Valid Statistic: Average Units: Percent
4xxErrorRate	The percentage of all requests for which the HTTP status code is 4xx. Valid Statistic: Average Units: Percent
5xxErrorRate	The percentage of all requests for which the HTTP status code is 5xx. Valid Statistic: Average Units: Percent

Dimensions for CloudFront Metrics

CloudFront metrics use the CloudFront namespace and provide metrics for two dimensions:

Dimension	Description
DistributionId	The CloudFront ID of the distribution for which you want to display metrics.
Region	The region for which you want to display metrics. This value must be Global.

DynamoDB Dimensions and Metrics

The metrics and dimensions that Amazon DynamoDB sends to Amazon CloudWatch are listed below. For more information about how to monitor Amazon DynamoDB, see [Monitoring DynamoDB Tables with Amazon CloudWatch](#) in the *Amazon DynamoDB Developer Guide*.

DynamoDB Metrics

The following metrics are available from DynamoDB. Note that DynamoDB only sends metrics to CloudWatch when they have a non-zero value. For example, the `UserErrors` metric is incremented whenever a request generates an HTTP 400 error code; if no requests have resulted in a 400 code during a particular time period, then no metrics for `UserErrors` are shown.

Note

Not all statistics, such as *Average* or *Sum*, are applicable for every metric. However, all of these values are available through the console, API, and command line client for all services. In the following table, each metric has a list of Valid Statistics that is applicable to that metric.

Metric	Description
SuccessfulRequestLatency	<p>The number of successful requests in the specified time period. By default, <code>SuccessfulRequestLatency</code> provides the elapsed time for successful calls. You can see statistics for the Minimum, Maximum, or Average, over time.</p> <p>Note CloudWatch also provides a Data Samples statistic: the total number of successful calls for a sample time period.</p> <p>Units: Milliseconds (or a count for Data Samples)</p> <p>Dimensions: <code>TableName</code>, <code>Operation</code></p> <p>Valid Statistics: Minimum, Maximum, Average, Data Samples</p>
UserErrors	<p>The number of requests generating a 400 status code (likely indicating a client error) response in the specified time period.</p> <p>Units: Count</p> <p>Dimensions: All dimensions</p> <p>Valid Statistics: Sum, Data Samples</p>
SystemErrors	<p>The number of requests generating a 500 status code (likely indicating a server error) response in the specified time period.</p> <p>Units: Count</p> <p>Dimensions: All dimensions</p> <p>Valid Statistics: Sum, Data Samples</p>
ThrottledRequests	<p>The number of user requests that exceeded the preset provisioned throughput limits in the specified time period.</p> <p><code>ThrottledRequests</code> is incremented by 1 if any event within a request exceeds a provisioned throughput limit. For example, if you update an item in a table with global secondary indexes, there are multiple events — a write to the table, and a write to each index. If one or more of these events are throttled, then <code>ThrottledRequests</code> is incremented by 1.</p> <p>To gain insight into which event is throttling a request, compare <code>ThrottledRequests</code> with the <code>ReadThrottleEvents</code> and <code>WriteThrottleEvents</code> for the table and its indexes.</p> <p>Units: Count</p> <p>Dimensions: <code>TableName</code></p> <p>Valid Statistics: Sum, Data Samples</p>

Metric	Description
ReadThrottleEvents	<p>The number of read events that exceeded the preset provisioned throughput limits in the specified time period.</p> <p>A single API request can result in multiple events. For example, a <code>BatchGetItem</code> that reads 10 items is processed as ten <code>GetItem</code> events. For each event, <code>ReadThrottleEvents</code> is incremented by 1 if that event is throttled. The <code>ThrottledRequests</code> metric for the entire <code>BatchGetItem</code> is not incremented unless <i>all ten</i> of the <code>GetItem</code> events are throttled.</p> <p>The <code>TableName</code> dimension returns the <code>ReadThrottleEvents</code> for the table, but not for any global secondary indexes. To view <code>ReadThrottleEvents</code> for a global secondary index, you must specify both <code>TableName</code> and <code>GlobalSecondaryIndex</code>.</p> <p>Units: Count</p> <p>Dimensions: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Valid Statistics: Sum, Data Samples</p>
WriteThrottleEvents	<p>The number of write events that exceeded the preset provisioned throughput limits in the specified time period.</p> <p>A single API request can result in multiple events. For example, a <code>PutItem</code> request on a table with three global secondary indexes would result in four events — the table write, and each of the three index writes. For each event, <code>WriteThrottleEvents</code> metric is incremented by 1 if that event is throttled. If any of the events are throttled, then <code>ThrottledRequests</code> is also incremented by 1.</p> <p>The <code>TableName</code> dimension returns the <code>WriteThrottleEvents</code> for the table, but not for any global secondary indexes. To view <code>WriteThrottleEvents</code> for a global secondary index, you must specify both <code>TableName</code> and <code>GlobalSecondaryIndex</code>.</p> <p>Units: Count</p> <p>Dimensions: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Valid Statistics: Sum, Data Samples</p>

Metric	Description
ProvisionedReadCapacityUnits	<p>The number of provisioned read capacity units for a table or a global secondary index.</p> <p>The <code>TableName</code> dimension returns the <code>ProvisionedReadCapacityUnits</code> for the table, but not for any global secondary indexes. To view <code>ProvisionedReadCapacityUnits</code> for a global secondary index, you must specify both <code>TableName</code> and <code>GlobalSecondaryIndex</code>.</p> <p>Units: Count</p> <p>Dimensions: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Valid Statistics: Minimum, Maximum, Average, Sum</p>
ProvisionedWriteCapacityUnits	<p>The number of provisioned write capacity units for a table or a global secondary index</p> <p>The <code>TableName</code> dimension returns the <code>ProvisionedWriteCapacityUnits</code> for the table, but not for any global secondary indexes. To view <code>ProvisionedWriteCapacityUnits</code> for a global secondary index, you must specify both <code>TableName</code> and <code>GlobalSecondaryIndex</code>.</p> <p>Units: Count</p> <p>Dimensions: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Valid Statistics: Minimum, Maximum, Average, Sum</p>
ConsumedReadCapacityUnits	<p>The number of read capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. You can retrieve the total consumed read capacity for a table and all of its global secondary indexes, or for a particular global secondary index. For more information, see Provisioned Throughput in Amazon DynamoDB.</p> <p>Note Use the <code>Sum</code> value to calculate the consumed throughput. For example, get the <code>Sum</code> value over a span of 5 minutes. Divide the <code>Sum</code> value by the number of seconds in 5 minutes (300) to get an average for the <code>ConsumedReadCapacityUnits</code> per second. (Note that such an average does not highlight any large spikes in read activity that take place during this period.) You can compare the calculated value to the provisioned throughput value you provide DynamoDB.</p> <p>Units: Count</p> <p>Dimensions: <code>TableName</code>, <code>GlobalSecondaryIndexName</code></p> <p>Valid Statistics: Minimum, Maximum, Average, Sum</p>

Metric	Description
ConsumedWriteCapacityUnits	<p>The number of write capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. You can retrieve the total consumed write capacity for a table and all of its global secondary indexes, or for a particular global secondary index. For more information, see Provisioned Throughput in Amazon DynamoDB.</p> <p>Note Use the Sum value to calculate the consumed throughput. For example, get the Sum value over a span of 5 minutes. Divide the Sum value by the number of seconds in 5 minutes (300) to get an average for the ConsumedWriteCapacityUnits per second. (Note that such an average does not highlight any large spikes in write activity that take place during this period.) You can compare the calculated value to the provisioned throughput value you provide DynamoDB.</p> <p>Units: Count Dimensions: TableName , GlobalSecondaryIndexName Valid Statistics: Minimum, Maximum, Average, Sum</p>
ReturnedItemCount	<p>The number of items returned by a Scan or Query operation.</p> <p>Units: Count Dimensions: TableName Valid Statistics: Minimum, Maximum, Average, Data Samples, Sum</p>

Dimensions for DynamoDB Metrics

The metrics for DynamoDB are qualified by the values for the account, table name, global secondary index name, or operation. You can use the CloudWatch console to retrieve DynamoDB data along any of the dimensions in the table below.

Dimension	Description
TableName	This dimension limits the data you request to a specific table. This value can be any table name for the current account.
GlobalSecondaryIndexName	This dimension limits the data you request to a global secondary index on a table. If you specify GlobalSecondaryIndexName, you must also specify TableName.

Dimension	Description
Operation	<p>The operation corresponds to the DynamoDB service API, and can be one of the following:</p> <ul style="list-style-type: none">• PutItem• DeleteItem• UpdateItem• GetItem• BatchGetItem• Scan• Query <p>For all of the operations in the current DynamoDB service API, see Operations in Amazon DynamoDB.</p>

Amazon ElastiCache Dimensions and Metrics

The metrics and dimensions that Amazon ElastiCache sends to Amazon CloudWatch are listed below. For more information about how to monitor Amazon ElastiCache, see [Viewing Cache Cluster and Cache Node Metrics](#) in the *Amazon ElastiCache User Guide*.

Topics

- [Dimensions for ElastiCache Metrics \(p. 186\)](#)
- [Host-Level Metrics \(p. 186\)](#)
- [Metrics for Memcached \(p. 187\)](#)
- [Metrics for Redis \(p. 189\)](#)

Dimensions for ElastiCache Metrics

All ElastiCache metrics use the "AWS/ElastiCache" namespace and provide metrics for a single dimension, the *CacheNodeId*, which is the automatically-generated identifier for each cache node in the cache cluster. You can find out what these values are for your cache nodes using the `DescribeCacheClusters` API or **elasticache-describe-cache-clusters** command line utility.

Each metric is published under a single set of dimensions. When retrieving metrics, you must supply both the `CacheClusterId` and `CacheNodeId` dimensions.

See Also

- [Host-Level Metrics \(p. 186\)](#)
- [Metrics for Memcached \(p. 187\)](#)
- [Metrics for Redis \(p. 189\)](#)

Host-Level Metrics

The following table lists host-level metrics provided by ElastiCache for individual cache nodes.

See Also

- Metrics for Memcached (p. 187)
- Metrics for Redis (p. 189)

Metric	Description	Unit
CPUUtilization	The percentage of CPU utilization.	Percent
SwapUsage	The amount of swap used on the host.	Bytes
FreeableMemory	The amount of free memory available on the host.	Bytes
NetworkBytesIn	The number of bytes the host has read from the network.	Bytes
NetworkBytesOut	The number of bytes the host has written to the network.	Bytes

Metrics for Memcached

The following table lists the metrics provided by ElasticCache that are derived from the Memcached *stats* command. Each metric is calculated at the cache node level.

For complete documentation of the Memcached *stats* command, go to <https://github.com/memcached/memcached/blob/master/doc/protocol.txt>.

See Also

- Host-Level Metrics (p. 186)

Metric	Description	Unit
BytesUsedForCacheItems	The number of bytes used to store cache items.	Bytes
BytesReadIntoMemcached	The number of bytes that have been read from the network by the cache node.	Bytes
BytesWrittenOutFromMemcached	The number of bytes that have been written to the network by the cache node.	Bytes
CasBadval	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
CasHits	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
CasMisses	The number of Cas requests the cache has received where the key requested was not found.	Count
CmdFlush	The number of flush commands the cache has received.	Count
CmdGet	The number of get commands the cache has received.	Count

Metric	Description	Unit
CmdSet	The number of set commands the cache has received.	Count
CurrConnections	A count of the number of connections connected to the cache at an instant in time. Note that due to the design of Memcached, this will always return a minimum count of 10.	Count
CurrItems	A count of the number of items currently stored in the cache.	Count
DecrHits	The number of decrement requests the cache has received where the requested key was found.	Count
DecrMisses	The number of decrement requests the cache has received where the requested key was not found.	Count
DeleteHits	The number of delete requests the cache has received where the requested key was found.	Count
DeleteMisses	The number of delete requests the cache has received where the requested key was not found.	Count
Evictions	The number of non-expired items the cache evicted to allow space for new writes.	Count
GetHits	The number of get requests the cache has received where the key requested was found.	Count
GetMisses	The number of get requests the cache has received where the key requested was not found.	Count
IncrHits	The number of increment requests the cache has received where the key requested was found.	Count
IncrMisses	The number of increment requests the cache has received where the key requested was not found.	Count
Reclaimed	The number of expired items the cache evicted to allow space for new writes.	Count

For Memcached 1.4.14, the following additional metrics are provided.

Metric	Description	Unit
BytesUsedForHash	The number of bytes currently used by hash tables.	Bytes
CmdConfigGet	The cumulative number of "config get" requests.	Count
CmdConfigSet	The cumulative number of "config set" requests.	Count
CmdTouch	The cumulative number of "touch" requests.	Count
CurrConfig	The current number of configurations stored.	Count
EvictedUnfetched	The number of valid items evicted from the least recently used cache (LRU) which were never touched after being set.	Count

Metric	Description	Unit
ExpiredUnfetched	The number of expired items reclaimed from the LRU which were never touched after being set.	Count
SlabsMoved	The total number of slab pages that have been moved.	Count
TouchHits	The number of keys that have been touched and were given a new expiration time.	Count
TouchMisses	The number of items that have been touched, but were not found.	Count

The following table describes the available calculated cache level metrics.

Metric	Description	Unit
NewConnections	The number of new connections the cache has received. This is derived from the memcached total_connections statistic by recording the change in total_connections across a period of time. This will always be at least 1, due to a connection reserved for a ElastiCache.	Count
NewItems	The number of new items the cache has stored. This is derived from the memcached total_items statistic by recording the change in total_items across a period of time.	Count
UnusedMemory	The amount of unused memory the cache can use to store items. This is derived from the memcached statistics limit_maxbytes and bytes by subtracting bytes from limit_maxbytes.	Bytes

Metrics for Redis

The following table lists the metrics provided by ElastiCache. With the exception of *ReplicationLag*, these metrics are derived from the Redis *info* command. Each metric is calculated at the cache node level.

For complete documentation of the Redis *info* command, go to <http://redis.io/commands/info>.

See Also

- [Host-Level Metrics \(p. 186\)](#)

Metric	Description	Unit
CurrConnections	The number of client connections, excluding connections from read replicas.	Count
Evictions	The number of keys that have been evicted due to the <i>maxmemory</i> limit.	Count
Reclaimed	The total number of key expiration events.	Count

Metric	Description	Unit
NewConnections	The total number of connections that have been accepted by the server during this period.	Count
BytesUsedForCache	The total number of bytes allocated by Redis.	Bytes
CacheHits	The number of successful key lookups.	Count
CacheMisses	The number of unsuccessful key lookups.	Count
ReplicationLag	This metric is only applicable for a cache node running as a read replica. It represents how far behind, in seconds, the replica is in applying changes from the primary cache cluster.	Seconds

These are aggregations of certain kinds of commands, derived from *info commandstats*:

Metric	Description	Unit
GetTypeCmds	The total number of <i>get</i> types of commands. This is derived from the Redis commandstats statistic by summing all of the <i>get</i> types of commands (<i>get</i> , <i>mget</i> , <i>hget</i> , etc.)	Count
SetTypeCmds	The total number of <i>set</i> types of commands. This is derived from the Redis commandstats statistic by summing all of the <i>set</i> types of commands (<i>set</i> , <i>hset</i> , etc.)	Count
KeyBasedCmds	The total number of commands that are key-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more keys.	Count
StringBasedCmds	The total number of commands that are string-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more strings.	Count
HashBasedCmds	The total number of commands that are hash-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more hashes.	Count
ListBasedCmds	The total number of commands that are list-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more lists.	Count
SetBasedCmds	The total number of commands that are set-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more sets.	Count
SortedSetBasedCmds	The total number of commands that are sorted set-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more sorted sets.	Count

Metric	Description	Unit
CurrItems	The number of items in the cache. This is derived from the Redis keyspace statistic, summing all of the keys in the entire keyspace.	Count

Amazon EBS Dimensions and Metrics

Amazon Elastic Block Store (Amazon EBS) sends data points to CloudWatch for several metrics. Amazon EBS Magnetic and General Purpose (SSD) volumes automatically send five-minute metrics to CloudWatch. Provisioned IOPS (SSD) volumes automatically send one-minute metrics to CloudWatch. For more information about how to monitor Amazon EBS, see [Monitoring the Status of Your Volumes](#) in the *Amazon EC2 User Guide for Linux Instances*.

Amazon EBS Metrics

You can use the Amazon CloudWatch `GetMetricStatistics` API to get any of the Amazon EBS volume metrics listed in the following table. Similar metrics are grouped together in the table, and the metrics in the first two rows are also available for the local stores on Amazon EC2 instances.

Metric	Description
VolumeReadBytes	The total number of bytes transferred in a specified period of time. Data is only reported to Amazon CloudWatch when the volume is active. If the volume is idle, no data is reported to Amazon CloudWatch.
VolumeWriteBytes	Units: Bytes
VolumeReadOps	The total number of I/O operations in a specified period of time.
VolumeWriteOps	<p>Note</p> <p>To calculate the average I/O operations per second (IOPS) for the period, divide the total operations in the period by the number of seconds in that period.</p> <p>Units: Count</p>
VolumeTotalReadTime	The total number of seconds spent by all operations that completed in a specified period of time. If multiple requests are submitted at the same time, this total could be greater than the length of the period. For example, for a period of 5 minutes (300 seconds): if 700 operations completed during that period, and each operation took 1 second, the value would be 700 seconds.
VolumeTotalWriteTime	Units: Seconds
VolumeIdleTime	The total number of seconds in a specified period of time when no read or write operations were submitted.
VolumeQueueLength	<p>Units: Seconds</p> <p>The number of read and write operation requests waiting to be completed in a specified period of time.</p> <p>Units: Count</p>

Metric	Description
VolumeThroughputPercentage	<p>Used with Provisioned IOPS (SSD) volumes only. The percentage of I/O operations per second (IOPS) delivered of the total IOPS provisioned for an Amazon EBS volume. Provisioned IOPS (SSD) volumes deliver within 10 percent of the provisioned IOPS performance 99.9 percent of the time over a given year.</p> <p>Note</p> <p>During a write, if there are no other pending I/O requests in a minute, the metric value will be 100 percent. Also, a volume's I/O performance may become degraded temporarily due to an action you have taken (e.g., creating a snapshot of a volume during peak usage, running the volume on a non-EBS-optimized instance, accessing data on the volume for the first time).</p> <p>Units: Percent</p>
VolumeConsumedReadWriteOps	<p>Used with Provisioned IOPS (SSD) volumes only. The total amount of read and write operations (normalized to 256K capacity units) consumed in a specified period of time.</p> <p>I/O operations that are smaller than 256K each count as 1 consumed IOPS. I/O operations that are larger than 256K are counted in 256K capacity units. For example, a 1024K I/O would count as 4 consumed IOPS.</p> <p>Units: Count</p>

Dimensions for Amazon EBS Metrics

The only dimension that Amazon EBS sends to CloudWatch is the Volume ID. This means that all available statistics are filtered by Volume ID.

Amazon Elastic Compute Cloud Dimensions and Metrics

This section discusses the metrics and dimensions that Amazon Elastic Compute Cloud (Amazon EC2) sends to CloudWatch, and describes how to enable detailed (one-minute) monitoring for an EC2 instance. For information about Auto Scaling group metrics, which pertain to Amazon EC2 instances that are within an Auto Scaling group, see [Auto Scaling Dimensions and Metrics \(p. 176\)](#).

CloudWatch offers basic (five-minute) monitoring for Amazon EC2 by default. To access detailed monitoring of Amazon EC2 instances, you must enable it. For more information about how to monitor Amazon EC2, see [Monitoring Your Instances with CloudWatch](#) in the *Amazon EC2 User Guide for Linux Instances*.

Topics

- [Amazon EC2 Metrics \(p. 192\)](#)
- [Dimensions for Amazon EC2 Metrics \(p. 195\)](#)

Amazon EC2 Metrics

The following metrics are available from each EC2 instance.

Metric	Description
CPUCreditUsage	<p>(Only valid for T2 instances) The number of CPU credits consumed during the specified period.</p> <p>This metric identifies the amount of time during which physical CPUs were used for processing instructions by virtual CPUs allocated to the instance.</p> <p>Note CPU Credit metrics are available at a 5 minute frequency.</p> <p>Units: Count</p>
CPUCreditBalance	<p>(Only valid for T2 instances) The number of CPU credits that an instance has accumulated.</p> <p>This metric is used to determine how long an instance can burst beyond its baseline performance level at a given rate.</p> <p>Note CPU Credit metrics are available at a 5 minute frequency.</p> <p>Units: Count</p>
CPUUtilization	<p>The percentage of allocated EC2 compute units that are currently in use on the instance. This metric identifies the processing power required to run an application upon a selected instance.</p> <p>Units: Percent</p>
DiskReadOps	<p>Completed read operations from all ephemeral disks available to the instance in a specified period of time. If your instance uses Amazon EBS volumes, see Amazon EBS Metrics (p. 191).</p> <p>Note To calculate the average I/O operations per second (IOPS) for the period, divide the total operations in the period by the number of seconds in that period.</p> <p>Units: Count</p>
DiskWriteOps	<p>Completed write operations to all ephemeral disks available to the instance in a specified period of time. If your instance uses Amazon EBS volumes, see Amazon EBS Metrics (p. 191).</p> <p>Note To calculate the average I/O operations per second (IOPS) for the period, divide the total operations in the period by the number of seconds in that period.</p> <p>Units: Count</p>
DiskReadBytes	<p>Bytes read from all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 191).)</p> <p>This metric is used to determine the volume of the data the application reads from the hard disk of the instance. This can be used to determine the speed of the application.</p> <p>Units: Bytes</p>

Metric	Description
DiskWriteBytes	<p>Bytes written to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 191).)</p> <p>This metric is used to determine the volume of the data the application writes onto the hard disk of the instance. This can be used to determine the speed of the application.</p> <p>Units: Bytes</p>
NetworkIn	<p>The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance.</p> <p>Units: Bytes</p>
NetworkOut	<p>The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance.</p> <p>Units: Bytes</p>
StatusCheckFailed	<p>A combination of StatusCheckFailed_Instance and StatusCheckFailed_System that reports if either of the status checks has failed. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure.</p> <p>Note Status check metrics are available at 1 minute frequency. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state.</p> <p>Units: Count</p>
StatusCheckFailed_Instance	<p>Reports whether the instance has passed the EC2 instance status check in the last minute. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure.</p> <p>Note Status check metrics are available at 1 minute frequency. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state.</p> <p>Units: Count</p>

Metric	Description
StatusCheckFailed_System	<p>Reports whether the instance has passed the EC2 system status check in the last minute. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure.</p> <p>Note Status check metrics are available at 1 minute frequency. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state.</p> <p>Units: Count</p>

Amazon CloudWatch data for a new EC2 instance typically becomes available within one minute of the end of the first period of time requested (the *aggregation period*) in the query. You can set the period—the length of time over which statistics are aggregated—with the Period parameter. For more information on periods, see [Periods \(p. 7\)](#).

You can use the currently available dimensions for EC2 instances (for example, `ImageId` or `InstanceType`) to refine the metrics returned. For information about the dimensions you can use with EC2, see [Dimensions for Amazon EC2 Metrics \(p. 195\)](#).

Dimensions for Amazon EC2 Metrics

If you're using Detailed Monitoring, you can filter the EC2 instance data using any of the dimensions in the following table.

Dimension	Description
<code>AutoScalingGroupName</code>	This dimension filters the data you request for all instances in a specified capacity group. An <i>AutoScalingGroup</i> is a collection of instances you define if you're using the Auto Scaling service. This dimension is available only for EC2 metrics when the instances are in such an <i>AutoScalingGroup</i> . Available for instances with Detailed or Basic Monitoring enabled.
<code>ImageId</code>	This dimension filters the data you request for all instances running this EC2 Amazon Machine Image (AMI). Available for instances with Detailed Monitoring enabled.
<code>InstanceId</code>	This dimension filters the data you request for the identified instance only. This helps you pinpoint an exact instance from which to monitor data. Available for instances with Detailed Monitoring enabled.
<code>InstanceType</code>	This dimension filters the data you request for all instances running with this specified instance type. This helps you categorize your data by the type of instance running. For example, you might compare data from an <code>m1.small</code> instance and an <code>m1.large</code> instance to determine which has the better business value for your application. Available for instances with Detailed Monitoring enabled.

Elastic Load Balancing Dimensions and Metrics

Topics

- [Elastic Load Balancing Metrics \(p. 196\)](#)
- [Dimensions for Elastic Load Balancing Metrics \(p. 197\)](#)

This section discusses the metrics and dimensions that Elastic Load Balancing sends to CloudWatch. CloudWatch provides detailed monitoring of Elastic Load Balancing by default. Unlike Amazon EC2, you do not need to specifically enable detailed monitoring. For more information about how to monitor Elastic Load Balancing, see [Monitor Your Load Balancer Using Amazon CloudWatch](#) in the *Elastic Load Balancing Developer Guide*.

Elastic Load Balancing Metrics

The following Elastic Load Balancing metrics are available from Amazon CloudWatch.

Elastic Load Balancing only reports when requests are flowing through the load balancer. If there are no requests or data for a given metric, the metric will not be reported to CloudWatch. If there are requests flowing through the load balancer, Elastic Load Balancing will measure and send metrics for that load balancer in 60-second intervals.

Note: The Statistic value available through Amazon CloudWatch, such as Min or Count are not always applicable to every metric. However, they are all available through the console, API, and the command line interface (CLI). For each metric, be aware of the Preferred Statistic for all the Elastic Load Balancing metrics to track useful information.

Metric	Description
HealthyHostCount	<p>The count of the number of healthy instances in each Availability Zone. Hosts are declared healthy if they meet the threshold for the number of consecutive health checks that are successful. Hosts that have failed more health checks than the value of the unhealthy threshold are considered unhealthy. If cross-zone is enabled, the count of the number of healthy instances is calculated for all Availability Zones.</p> <p>Preferred statistic: average</p>
UnHealthyHostCount	<p>The count of the number of unhealthy instances in each Availability Zone. Hosts that have failed more health checks than the value of the unhealthy threshold are considered unhealthy. If cross-zone is enabled, the count of the number of unhealthy instances is calculated for all Availability Zones. Instances may become unhealthy due to connectivity issues, health checks returning non-200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check.</p> <p>Preferred statistic: average</p>
RequestCount	<p>The count of the number of completed requests that were received and routed to the back-end instances.</p> <p>Preferred statistic: sum</p>
Latency	<p>Measures the time elapsed in seconds after the request leaves the load balancer until the response is received.</p> <p>Preferred statistic: average</p>

Metric	Description
HTTPCode_ELB_4XX	The count of the number of HTTP 4XX client error codes generated by the load balancer when the listener is configured to use HTTP or HTTPS protocols. Client errors are generated when a request is malformed or is incomplete. Preferred statistic: sum
HTTPCode_ELB_5XX	The count of the number of HTTP 5XX server error codes generated by the load balancer when the listener is configured to use HTTP or HTTPS protocols. This metric does not include any responses generated by back-end instances. The metric is reported if there are no back-end instances that are healthy or registered to the load balancer, or if the request rate exceeds the capacity of the instances or the load balancers. Preferred statistic: sum
HTTPCode_Backend_2XX HTTPCode_Backend_3XX HTTPCode_Backend_4XX HTTPCode_Backend_5XX	The count of the number of HTTP response codes generated by back-end instances. This metric does not include any response codes generated by the load balancer. The 2XX class status codes represent successful actions. The 3XX class status code indicates that the user agent requires action. The 4XX class status code represents client errors. The 5XX class status code represents back-end server errors. Preferred statistic: sum
BackendConnectionErrors	The count of the number of connections that were not successfully established between the load balancer and the registered instances. Because the load balancer will retry when there are connection errors, this count can exceed the request rate. Preferred statistic: sum
SurgeQueueLength	A count of the total number of requests that are pending submission to a registered instance. Preferred statistic: max
SpilloverCount	A count of the total number of requests that were rejected due to the queue being full. Preferred statistic: sum

Dimensions for Elastic Load Balancing Metrics

You can use the currently available dimensions for Elastic Load Balancing to refine the metrics returned by a query. For example, you could use *HealthyHostCount* and dimensions *LoadBalancerName* and *AvailabilityZone* to get the Average number of healthy Instances behind the specified LoadBalancer within the specified Availability Zone for a given period of time.

Elastic Load Balancing data can be aggregated along any of the following dimensions shown in the table below.

Dimension	Description
LoadBalancerName	Limits the metric data to Amazon EC2 instances that are connected to the specified load balancer.

Dimension	Description
AvailabilityZone	Limits the metric data to load balancers in the specified <i>Availability Zone</i> .

Amazon Elastic MapReduce Dimensions and Metrics

This section discusses the metrics and dimensions that Amazon Elastic MapReduce (Amazon EMR) sends to CloudWatch. All Amazon EMR job flows automatically send metrics in five-minute intervals. Metrics are archived for two weeks; after that period, the data is discarded. For more information about how to monitor Amazon EMR, see [Monitor Metrics with Amazon CloudWatch](#) in the *Amazon Elastic MapReduce Developer Guide*.

Amazon EMR Metrics

Amazon EMR sends the following metrics to Amazon CloudWatch.

Note

Amazon EMR pulls metrics from a cluster. If a cluster becomes unreachable, no metrics will be reported until the cluster becomes available again.

Metric	Description
CoreNodesPending	The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: Count
CoreNodesRunning	The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: Count
HBaseBackupFailed	Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: Count

Metric	Description
HBaseMostRecentBackupDuration	<p>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes since the backup started. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase Backups</p> <p>Units: Minutes</p>
HBaseTimeSinceLastSuccessfulBackup	<p>The number of elapsed minutes since the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: Minutes</p>
HDFSBytesRead	<p>The number of bytes read from HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: Count</p>
HDFSBytesWritten	<p>The number of bytes written to HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: Count</p>
HDFSUtilization	<p>The percentage of HDFS storage currently used.</p> <p>Use case: Analyze cluster performance</p> <p>Units: Percent</p>
IsIdle	<p>Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.</p> <p>Use case: Monitor cluster performance</p> <p>Units: Count</p>
JobsFailed	<p>The number of jobs in the cluster that have failed.</p> <p>Use case: Monitor cluster health</p> <p>Units: Count</p>

Metric	Description
JobsRunning	<p>The number of jobs in the cluster that are currently running.</p> <p>Use case: Monitor cluster health</p> <p>Units: Count</p>
LiveDataNodes	<p>The percentage of data nodes that are receiving work from Hadoop.</p> <p>Use case: Monitor cluster health</p> <p>Units: Percent</p>
LiveTaskTrackers	<p>The percentage of task trackers that are functional.</p> <p>Use case: Monitor cluster health</p> <p>Units: Percent</p>
MapSlotsOpen	<p>The unused map task capacity. This is calculated as the maximum number of map tasks for a given cluster, less the total number of map tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: Count</p>
MissingBlocks	<p>The number of blocks in which HDFS has no replicas. These might be corrupt blocks.</p> <p>Use case: Monitor cluster health</p> <p>Units: Count</p>
ReduceSlotsOpen	<p>Unused reduce task capacity. This is calculated as the maximum reduce task capacity for a given cluster, less the number of reduce tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: Count</p>
RemainingMapTasks	<p>The number of remaining map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. A remaining map task is one that is not in any of the following states: Running, Killed, or Completed.</p> <p>Use case: Monitor cluster progress</p> <p>Units: Count</p>
RemainingMapTasksPerSlot	<p>The ratio of the total map tasks remaining to the total map slots available in the cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: Ratio</p>

Metric	Description
RemainingReduceTasks	<p>The number of remaining reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: Count</p>
RunningMapTasks	<p>The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs will be generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: Count</p>
RunningReduceTasks	<p>The number of running reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: Count</p>
S3BytesRead	<p>The number of bytes read from Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: Count</p>
S3BytesWritten	<p>The number of bytes written to Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: Count</p>
TaskNodesPending	<p>The number of core nodes waiting to be assigned. All of the task nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: Count</p>
TaskNodesRunning	<p>The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: Count</p>
TotalLoad	<p>The total number of concurrent data transfers.</p> <p>Use case: Monitor cluster health</p> <p>Units: Count</p>

Amazon EMR Dimensions

The following dimensions are available for Amazon EMR.

Dimension	Description
ClusterId	The identifier for a cluster. You can find this value by clicking on the cluster in the Amazon EMR console. It takes the form j-XXXXXXXXXXXXXX.
JobId	The identifier of a job within a cluster. You can use this to filter the metrics returned from a cluster down to those that apply to a single job within the cluster. JobId takes the form job_XXXXXXXXXXXX_XXXX.

Amazon Kinesis Dimensions and Metrics

Amazon Kinesis sends metrics and dimensions for your streams to CloudWatch. For more information about how to monitor Amazon Kinesis, see [Monitoring Amazon Kinesis with Amazon CloudWatch](#) in the *Amazon Kinesis Developer Guide*.

Topics

- [Amazon Kinesis Metrics \(p. 202\)](#)
- [Dimensions for Amazon Kinesis Metrics \(p. 203\)](#)

Amazon Kinesis Metrics

Amazon Kinesis sends the following metrics to CloudWatch.

Metric	Description
PutRecord.Bytes	The number of bytes put to the Amazon Kinesis stream over the specified time period. Units: Bytes
PutRecord.Latency	The time taken per PutRecord operation, measured over the specified time period. Units: Milliseconds
PutRecord.Success	The number of successful PutRecord operations per Amazon Kinesis stream, measured over the specified time period. Units: Count
GetRecords.Bytes	The number of bytes retrieved from the stream, measured over the specified time period. Units: Bytes

Metric	Description
GetRecords.IteratorAgeInMilliseconds	The age of the iterator used to request records for all shards in your stream, measured over the specified time period. Units: Milliseconds
GetRecords.Latency	The time taken per GetRecords operation, measured over the specified time period. Units: Milliseconds
GetRecords.Success	The number of successful GetRecords operations per stream, measured over the specified time period. Units: Count

Dimensions for Amazon Kinesis Metrics

You can use the following dimensions to refine the metrics for Amazon Kinesis.

Dimension	Description
StreamName	The name of the Amazon Kinesis stream. All available statistics are filtered by StreamName.

AWS OpsWorks Dimensions and Metrics

This section discusses the metrics and dimensions that AWS OpsWorks sends to CloudWatch. CloudWatch provides detailed monitoring of AWS OpsWorks by default. AWS OpsWorks sends metrics for each active stack every minute. Unlike Amazon EC2 and Auto Scaling, you do not need to specifically enable detailed monitoring. For more information about how to monitor AWS OpsWorks, see [Monitoring](#) in the *AWS OpsWorks User Guide*.

Topics

- [AWS OpsWorks Metrics \(p. 203\)](#)
- [Dimensions for AWS OpsWorks Metrics \(p. 204\)](#)

AWS OpsWorks Metrics

The following metrics are available from AWS OpsWorks.

Metric	Description
cpu_idle	The percentage of time that the CPU is idle. Units: Percent
cpu_nice	The percentage of time that the CPU is handling processes with a positive nice value, which have lower scheduling priority. For information, see nice (Unix) . Units: Percent

Metric	Description
cpu_system	The percentage of time that the CPU is handling system operations. Units: Percent
cpu_user	The percentage of time that the CPU is handling user operations. Units: Percent
cpu_waitio	The percentage of time that the CPU is waiting for input/output operations. Units: Percent
load_1	The load averaged over a 1-minute window. Units: Unix load units
load_5	The load averaged over a 5-minute window. Units: Unix load units
load_15	The load averaged over a 15-minute window. Units: Unix load units
memory_buffers	The amount of buffered memory. Units: Kilobytes
memory_cached	The amount of cached memory. Units: Kilobytes
memory_free	The amount of free memory. Units: Kilobytes
memory_swap	The amount of swap space. Units: Kilobytes
memory_total	The total amount of memory. Units: Kilobytes
memory_used	The amount of memory in use. Units: Kilobytes
procs	The number of active processes. Units: Count

Dimensions for AWS OpsWorks Metrics

AWS OpsWorks data can be filtered along any of the following dimensions in the table below.

Dimension	Description
StackId	Average values for a stack.
LayerId	Average values for a layer.
InstanceId	Average values for an instance.

Amazon Redshift Dimensions and Metrics

This section discusses the metrics and dimensions that Amazon Redshift sends to CloudWatch. CloudWatch provides detailed monitoring of Amazon Redshift by default. Amazon Redshift sends metrics for each active cluster every minute. Unlike Amazon EC2 and Auto Scaling, you do not need to specifically enable detailed monitoring. For more information about how to monitor Amazon Redshift, see [Monitoring Amazon Redshift Cluster Performance](#) in the *Amazon Redshift Cluster Management Guide*.

Topics

- [Amazon Redshift Metrics \(p. 205\)](#)
- [Dimensions for Amazon Redshift Metrics \(p. 207\)](#)

Amazon Redshift Metrics

The following metrics are available from Amazon Redshift.

Metric	Description
CPUUtilization	The percentage of CPU utilization. For clusters, this metric represents an aggregation of all nodes (leader and compute) CPU utilization values. Units: Percent Dimensions: node (leader and compute), cluster
DatabaseConnections	The number of database connections to a cluster. Units: Count Dimensions: cluster

Metric	Description
HealthStatus	<p>Indicates the health of the cluster. Every minute the cluster connects to its database and performs a simple query. If it is able to perform this operation successfully, the cluster is considered healthy. Otherwise, the cluster is unhealthy. An unhealthy status can occur when the cluster database is under extremely heavy load or if there is a configuration problem with a database on the cluster. The exception to this is when the cluster is undergoing maintenance. Even though your cluster might be unavailable due to maintenance tasks, the cluster remains in HEALTHY state. For more information, see Maintenance Windows in the <i>Amazon Redshift Cluster Management Guide</i>.</p> <p>Note In Amazon CloudWatch this metric is reported as 1 or 0 whereas in the Amazon CloudWatch console, this metric is displayed with the words HEALTHY or UNHEALTHY for convenience. When this metric is displayed in the Amazon CloudWatch console, sampling averages are ignored and only HEALTHY or UNHEALTHY are displayed. In Amazon CloudWatch, values different than 1 and 0 may occur because of sampling issue. Any value below 1 for <code>HealthStatus</code> is reported as 0 (UNHEALTHY).</p> <p>Units: 1/0 (HEALTHY/UNHEALTHY in the Amazon CloudWatch console) Dimensions: cluster</p>
MaintenanceMode	<p>Indicates whether the cluster is in maintenance mode.</p> <p>Note In Amazon CloudWatch this metric is reported as 1 or 0 whereas in the Amazon CloudWatch console, this metric is displayed with the words ON or OFF for convenience. When this metric is displayed in the Amazon CloudWatch console, sampling averages are ignored and only ON or OFF are displayed. In Amazon CloudWatch, values different than 1 and 0 may occur because of sampling issues. Any value greater than 0 for <code>MaintenanceMode</code> is reported as 1 (ON).</p> <p>Units: 1/0 (ON/OFF in the Amazon CloudWatch console). Dimensions: cluster</p>
NetworkReceiveThroughput	<p>The rate at which the node or cluster receives data.</p> <p>Units: Bytes/seconds (MB/s in the Amazon CloudWatch console) Dimensions: node (leader and compute), cluster</p>
NetworkTransmitThroughput	<p>The rate at which the node or cluster writes data.</p> <p>Units: Bytes/second (MB/s in the Amazon CloudWatch console) Dimensions: node (leader and compute), cluster</p>
PercentageDiskSpaceUsed	<p>The percent of disk space used.</p> <p>Units: Percent Dimensions: node, cluster</p>

Metric	Description
ReadIOPS	The average number of disk read operations per second. Units: Count/second Dimensions: node
ReadLatency	The average amount of time taken for disk read I/O operations. Units: Seconds Dimensions: node
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes (GB/s in the Amazon CloudWatch console) Dimensions: node
WriteIOPS	The average number of write operations per second. Units: Count/seconds Dimensions: node
WriteLatency	The average amount of time taken for disk write I/O operations. Units: Seconds Dimensions: node
WriteThroughput	The average number of bytes written to disk per second. Units: Bytes (GB/s in the Amazon CloudWatch console) Dimensions: node

Dimensions for Amazon Redshift Metrics

Amazon Redshift data can be filtered along any of the following dimensions in the table below.

Dimension	Description
NodeID	Filters requested data that is specific to the nodes of a cluster. NodeID will be either "Leader", "Shared", or "Compute-N" where N is 0, 1, ... for the number of nodes in the cluster. "Shared" means that the cluster has only one node, i.e. the leader node and compute node are combined.
ClusterIdentifier	Filters requested data that is specific to the cluster. Metrics that are specific to clusters include HealthStatus, MaintenanceMode, and DatabaseConnections. In general metrics in for this dimension (e.g. ReadIOPS) that are also metrics of nodes represent an aggregate of the node metric data. You should take care in interpreting these metrics because they aggregate behavior of leader and compute nodes.

Amazon RDS Dimensions and Metrics

This section discusses the metrics and dimensions that Amazon Relational Database Service sends to CloudWatch. CloudWatch provides detailed monitoring of Amazon RDS by default. Amazon Relational Database Service sends metrics for each active database instance every minute. Unlike Amazon EC2 and Auto Scaling, you do not need to specifically enable detailed monitoring. For more information about how to monitor Amazon RDS, see [Monitoring a DB Instance](#) in the *Amazon Relational Database Service User Guide*.

Topics

- [Amazon RDS Metrics \(p. 208\)](#)
- [Dimensions for RDS Metrics \(p. 209\)](#)

Amazon RDS Metrics

The following metrics are available from Amazon Relational Database Service.

Metric	Description
BinLogDiskUsage	The amount of disk space occupied by binary logs on the master. Applies to MySQL read replicas. Units: Bytes
CPUUtilization	The percentage of CPU utilization. Units: Percent
DatabaseConnections	The number of database connections in use. Units: Count
DiskQueueDepth	The number of outstanding IOs (read/write requests) waiting to access the disk. Units: Count
FreeableMemory	The amount of available random access memory. Units: Bytes
FreeStorageSpace	The amount of available storage space. Units: Bytes
ReplicaLag	The amount of time a Read Replica DB Instance lags behind the source DB Instance. Applies to MySQL read replicas. The ReplicaLag metric reports the value of the Seconds_Behind_Master field of the MySQL SHOW SLAVE STATUS command. For more information, see SHOW SLAVE STATUS . Units: Seconds
SwapUsage	The amount of swap space used on the DB Instance. Units: Bytes

Metric	Description
ReadIOPS	The average number of disk I/O operations per second. Units: Count/Second
WriteIOPS	The average number of disk I/O operations per second. Units: Count/Second
ReadLatency	The average amount of time taken per disk I/O operation. Units: Seconds
WriteLatency	The average amount of time taken per disk I/O operation. Units: Seconds
ReadThroughput	The average number of bytes read from disk per second. Units: Bytes/Second
WriteThroughput	The average number of bytes written to disk per second. Units: Bytes/Second
NetworkReceiveThroughput	The incoming (Receive) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes
NetworkTransmitThroughput	The outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication. Units: Bytes

Dimensions for RDS Metrics

Amazon RDS data can be filtered along any of the following dimensions in the table below.

Dimension	Description
DBInstanceIdentifier	This dimension filters the data you request for a specific database instance.
DatabaseClass	This dimension filters the data you request for all instances in a database class. For example, you can aggregate metrics for all instances that belong to the database class db.m1.small
EngineName	This dimension filters the data you request for the identified engine name only. For example, you can aggregate metrics for all instances that have the engine name mysql.

Amazon Route 53 Dimensions and Metrics

This section discusses the metrics and dimensions that Amazon Route 53 sends to CloudWatch. CloudWatch provides detailed monitoring of Amazon Route 53 by default. Amazon Route 53 sends one-minute metrics to CloudWatch. For more information about how to monitor Amazon Route 53, see [Monitoring Health Checks Using Amazon CloudWatch](#) in the *Amazon Route 53 Developer Guide*.

Amazon Route 53 Metrics

Metric	Description
HealthCheckStatus	The status of the health check endpoint that CloudWatch is checking. 1 indicates healthy, and 0 indicates unhealthy. Valid Statistics: Minimum Units: none
HealthCheckPercentageHealthy	The percentage of Amazon Route 53 health checkers that consider the selected endpoint to be healthy. Valid Statistics: Minimum, Maximum, Average Units: Percent

Dimensions for Amazon Route 53 Metrics

Amazon Route 53 metrics use the AWS/Route53 namespace and provide metrics for a single dimension, `HealthCheckId`, which is the automatically generated identifier for each health check. When retrieving metrics, you must supply the `HealthCheckId` dimension.

For more information, see [Monitoring Health Checks Using CloudWatch](#) in the *Amazon Route 53 Developer Guide*.

Amazon Simple Notification Service Dimensions and Metrics

Amazon Simple Notification Service sends data points to CloudWatch for several metrics. All active topics automatically send five-minute metrics to CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon Simple Notification Service. A topic stays active for six hours from the last activity (i.e. any API call) on the topic. For more information about how to monitor Amazon SNS, see [Monitoring Amazon SNS with Amazon CloudWatch](#) in the *Amazon Simple Notification Service Developer Guide*.

Amazon Simple Notification Service Metrics

This section discusses the metrics that Amazon Simple Notification Service (Amazon SNS) sends to CloudWatch.

Metric	Description
NumberOfMessagesPublished	The number of messages published. Units: Count Valid Statistics: Sum
PublishSize	The size of messages published. Units: Bytes Valid Statistics: Minimum, Maximum, Average and Count
NumberOfNotificationsDelivered	The number of messages successfully delivered. Units: Count Valid Statistics: Sum
NumberOfNotificationsFailed	The number of messages that SNS failed to deliver. Units: Count Valid Statistics: Sum

Dimensions for Amazon Simple Notification Service Metrics

Amazon Simple Notification Service sends the following dimensions to CloudWatch.

Dimension	Description
Application	Filters on application objects, which represent an app and device registered with one of the supported push notification services, such as APNS and GCM.
Application,Platform	Filters on application and platform objects, where the platform objects are for the supported push notification services, such as APNS and GCM.
Platform	Filters on platform objects for the push notification services, such as APNS and GCM.
TopicName	Filters on Amazon SNS topic names.

Amazon SQS Dimensions and Metrics

Amazon SQS sends data points to CloudWatch for several metrics. All active queues automatically send five-minute metrics to CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon SQS. A queue stays active for six hours from the last activity (i.e. any API call) on the queue. For more information about how to monitor Amazon SQS, see [Monitoring Amazon SQS with Amazon CloudWatch](#) in the *Amazon Simple Queue Service Developer Guide*.

Amazon SQS Metrics

This section discusses the metrics that Amazon Simple Queue Service (Amazon SQS) sends to CloudWatch.

Metric	Description
NumberOfMessagesSent	The number of messages added to a queue. Units: Count Valid Statistics: Sum
SentMessageSize	The size of messages added to a queue. Units: Bytes Valid Statistics: Minimum, Maximum, Average and Count
NumberOfMessagesReceived	The number of messages returned by calls to the <code>ReceiveMessage</code> API action. Units: Count Valid Statistics: Sum
NumberOfEmptyReceives	The number of <code>ReceiveMessage</code> API calls that did not return a message. Units: Count Valid Statistics: Sum
NumberOfMessagesDeleted	The number of messages deleted from the queue. Units: Count Valid Statistics: Sum
ApproximateNumberOfMessagesDelayed	The number of messages in the queue that are delayed and not available for reading immediately. This can happen when the queue is configured as a delay queue or when a message has been sent with a delay parameter. Units: Count Valid Statistics: Average
ApproximateNumberOfMessagesVisible	The number of messages available for retrieval from the queue. Units: Count Valid Statistics: Average

Metric	Description
ApproximateNumberOfMessagesNotVisible	<p>The number of messages that are in flight. Messages are considered in flight if they have been sent to a client but have not yet been deleted or have not yet reached the end of their visibility window.</p> <p>Units: Count</p> <p>Valid Statistics: Average</p>

Dimensions for Amazon SQS Metrics

The only dimension that Amazon SQS sends to CloudWatch is QueueName. This means that all available statistics are filtered by QueueName.

Amazon SWF Dimensions and Metrics

Amazon SWF sends data points to CloudWatch for several metrics. Some of the Amazon SWF metrics for CloudWatch are time intervals, always measured in milliseconds. These metrics generally correspond to stages of your workflow execution for which you can set workflow and activity timeouts, and have similar names. For example, the **DecisionTaskStartToCloseTime** metric measures the time it took for the decision task to complete after it began executing, which is the same time period for which you can set a **DecisionTaskStartToCloseTimeout** value.

Other Amazon SWF metrics report results as a count. For example, **WorkflowsCanceled**, records a result as either one or zero, indicating whether or not the workflow was canceled. A value of zero does not indicate that the metric was not reported, only that the condition described by the metric did not occur. For count metrics, minimum and maximum will always be either zero or one, but average will be a value ranging from zero to one. For more information about how to monitor Amazon SWF, see [Viewing Amazon SWF Metrics for CloudWatch using the AWS Management Console](#); in the *Amazon Simple Workflow Service Developer Guide*.

Workflow Metrics

Dimensions for Amazon SWF workflow metrics include the **Domain**, **WorkflowTypeName**, **WorkflowTypeVersion** and **Metric Name**.

The following metrics are available for Amazon SWF workflows:

- **DecisionTaskScheduleToStartTime** – the time interval, in milliseconds, between the time that the decision task was scheduled and the time it was picked up by a worker and started.
- **DecisionTaskStartToCloseTime** – the time interval, in milliseconds, between the time that the decision task was started and the time it was closed.
- **DecisionTasksCompleted** – the count of decision tasks that have been completed.
- **StartedDecisionTasksTimedOutOnClose** – the count of decision tasks that started but timed out on closing.
- **WorkflowStartToCloseTime** – the time, in milliseconds, between the time the workflow started and the time it closed.
- **WorkflowsCanceled** – the count of workflows that were canceled.
- **WorkflowsCompleted** – the count of workflows that completed.

- **WorkflowsContinuedAsNew** – the count of workflows that continued as new.
- **WorkflowsFailed** – the count of workflows that failed.
- **WorkflowsTerminated** – the count of workflows that were terminated.
- **WorkflowsTimedOut** – the count of workflows that timed out, for any reason.

Activity Metrics

Dimensions for Amazon SWF activity metrics include the **Domain**, **ActivityTypeName**, **ActivityTypeVersion** and **Metric Name**.

The following metrics are available for Amazon SWF activities:

- **ActivityTaskScheduleToCloseTime** – the time interval, in milliseconds, between the time when the activity was scheduled to when it closed.
- **ActivityTaskScheduleToStartTime** – the time interval, in milliseconds, between the time when the activity task was scheduled and when it started.
- **ActivityTaskStartToCloseTime** – the time interval, in milliseconds, between the time when the activity task started and when it was closed.
- **ActivityTasksCanceled** – the count of activity tasks that were canceled.
- **ActivityTasksCompleted** – the count of activity tasks that completed.
- **ActivityTasksFailed** – the count of activity tasks that failed.
- **ScheduledActivityTasksTimedOutOnClose** – the count of activity tasks that were scheduled but timed out on close.
- **ScheduledActivityTasksTimedOutOnStart** – the count of activity tasks that were scheduled but timed out on start.
- **StartedActivityTasksTimedOutOnClose** – the count of activity tasks that were started but timed out on close.
- **StartedActivityTasksTimedOutOnHeartbeat** – the count of activity tasks that were started but timed out due to a heartbeat timeout.

AWS Storage Gateway Dimensions and Metrics

AWS Storage Gateway sends data points to CloudWatch for several metrics. All active queues automatically send five-minute metrics to CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for AWS Storage Gateway. For more information about how to monitor AWS Storage Gateway, see [Monitoring Your AWS Storage Gateway](#) in the *AWS Storage Gateway User Guide*.

AWS Storage Gateway Metrics

The following metrics are available from the AWS Storage Gateway Service.

The following table describes the AWS Storage Gateway metrics that you can use to get information about your gateways. Specify the `GatewayId` or `GatewayName` dimension for each metric to view the data for a gateway. Note that these metrics are measured in 5-minute intervals.

Metric	Description
ReadBytes	<p>The total number of bytes read from your on-premises applications in the reporting period for all volumes in the gateway.</p> <p>Use this metric to measure throughput by selecting the <code>Sum</code> statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the <code>Samples</code> statistic and dividing each data point by the Period.</p> <p>Units: Bytes</p>
WriteBytes	<p>The total number of bytes written to your on-premises applications in the reporting period for all volumes in the gateway.</p> <p>Use this metric to measure throughput by selecting the <code>Sum</code> statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the <code>Samples</code> statistic and dividing each data point by the Period.</p> <p>Units: Bytes</p>
ReadTime	<p>The total number of milliseconds spent to do reads from your on-premises applications in the reporting period for all volumes in the gateway.</p> <p>Use this metric with the <code>Average</code> statistic to measure average latency.</p> <p>Units: Milliseconds</p>
WriteTime	<p>The total number of milliseconds spent to do writes from your on-premises applications in the reporting period for all volumes in the gateway.</p> <p>Use this metric with the <code>Average</code> statistic to measure average latency.</p> <p>Units: Milliseconds</p>
QueuedWrites	<p>The number of bytes waiting to be written to AWS, sampled at the end of the reporting period for all volumes in the gateway. These bytes are kept in your gateway's working storage.</p> <p>Units: Bytes</p>
CloudBytesDownloaded	<p>The total number of bytes that the gateway downloaded from AWS during the reporting period.</p> <p>Use this metric to measure throughput by selecting the <code>Sum</code> statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the <code>Samples</code> statistic and dividing each data point by the Period.</p> <p>Units: Bytes</p>
CloudBytesUploaded	<p>The total number of bytes that the gateway uploaded to AWS during the reporting period.</p> <p>Use this metric to measure throughput by selecting the <code>Sum</code> statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the <code>Samples</code> statistic and dividing each data point by the Period.</p> <p>Units: Bytes</p>

Metric	Description
CloudDownloadLatency	<p>The total number of milliseconds spent reading data from AWS during the reporting period.</p> <p>Use this metric with the <code>Average</code> statistic to measure average latency.</p> <p>Units: Milliseconds</p>
WorkingStoragePercentUsed	<p>Percent utilization of the gateway's working storage. The sample is taken at the end of the reporting period.</p> <p>Units: Percent</p>
WorkingStorageUsed	<p>The total number of bytes being used in the gateway's working storage. The sample is taken at the end of the reporting period.</p> <p>Units: Bytes</p>
WorkingStorageFree	<p>The total amount of unused space in the gateway's working storage. The sample is taken at the end of the reporting period.</p> <p>Units: Bytes</p>

The following table describes the AWS Storage Gateway metrics that you can use to get information about your storage volumes. Specify the `VolumeId` dimension for each metric to view the data for a storage volume.

Metric	Description
ReadBytes	<p>The total number of bytes read from your on-premises applications in the reporting period.</p> <p>Use this metric to measure throughput by selecting the <code>Sum</code> statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the <code>Samples</code> statistic and dividing each data point by the Period.</p> <p>Units: Bytes</p>
WriteBytes	<p>The total number of bytes written to your on-premises applications in the reporting period.</p> <p>Use this metric to measure throughput by selecting the <code>Sum</code> statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the <code>Samples</code> statistic and dividing each data point by the Period.</p> <p>Units: Bytes</p>
ReadTime	<p>The total number of milliseconds spent to do reads from your on-premises applications in the reporting period.</p> <p>Use this metric with the <code>Average</code> statistic to measure average latency.</p> <p>Units: Milliseconds</p>

Metric	Description
WriteTime	<p>The total number of milliseconds spent to do writes from your on-premises applications in the reporting period.</p> <p>Use this metric with the <code>Average</code> statistic to measure average latency.</p> <p>Units: Milliseconds</p>
QueuedWrites	<p>The number of bytes waiting to be written to AWS, sampled at the end of the reporting period.</p> <p>Units: Bytes</p>
CacheHitPercent	<p>Percent of application reads served from the cache. This metric applies only to the gateway-cached volume setup. The sample is taken at the end of the reporting period.</p> <p>Units: Percent.</p>
CachePercentUsed	<p>Percent utilization of the gateway cache storage. This metric applies only to the gateway-cached volume setup. The sample is taken at the end of the reporting period.</p> <p>Units: Percent</p>
CachePercentDirty	<p>Percent of the gateway cache that has not been persisted to AWS. This metric applies only to the gateway-cached volume setup. The sample is taken at the end of the reporting period.</p> <p>Units: Percent</p>
UploadBufferFree	<p>The total amount of unused space in the gateway upload buffer. The sample is taken at the end of the reporting period.</p> <p>Units: Bytes</p>
UploadBufferPercentUsed	<p>Percent utilization of the gateway upload buffer. The sample is taken at the end of the reporting period.</p> <p>Units: Percent</p>
UploadBufferUsed	<p>The total number of bytes being used in the gateway upload buffer. The sample is taken at the end of the reporting period.</p> <p>Units: Bytes</p>
TotalCacheSize	<p>The total size of the cache in bytes. This metric applies only to the gateway-cached volume setup. The sample is taken at the end of the reporting period.</p> <p>Units: Bytes</p>

Metric	Description
WorkingStoragePercentUsed	<p>Percent utilization of the gateway upload buffer. The sample is taken at the end of the reporting period.</p> <p>Note Working storage applies only to the gateway-stored volume setup. The upload buffer applies to both the gateway-stored and gateway-cached volume setups. If you are working with both types of gateway setups, you may find it more convenient to use just the corresponding upload buffer metric, UploadBufferPercentUsed.</p> <p>Units: Percent</p>

Dimensions for AWS Storage Gateway Metrics

The Amazon CloudWatch namespace for the AWS Storage Gateway service is `AWS/StorageGateway`. Data is available automatically in 5-minute periods at no charge.

Dimension	Description
GatewayId, GatewayName	<p>These dimensions filter the data you request to gateway-specific metrics. You can identify a gateway to work by its <code>GatewayId</code> or its <code>GatewayName</code>. However, note that if the name of your gateway was changed for the time range that you are interested in viewing metrics, then you should use the <code>GatewayId</code>.</p> <p>Throughput and latency data of a gateway is based on all the volumes for the gateway. For information about working with gateway metrics, see Measuring Performance Between Your Gateway and AWS.</p>
VolumeId	<p>This dimension filters the data you request to volume-specific metrics. Identify a storage volume to work with by its <code>VolumeId</code>. For information about working with volume metrics, see Measuring Performance Between Your Application and Gateway.</p>

Making API Requests

Query requests used with Amazon CloudWatch are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs.

Topics

- [Amazon CloudWatch Endpoints \(p. 219\)](#)
- [Query Parameters \(p. 219\)](#)
- [The RequestId \(p. 220\)](#)
- [Query API Authentication \(p. 220\)](#)
- [Query API Examples Using Signature Version 2 \(p. 221\)](#)
- [Query API Error Messages Using Signature Version 2 \(p. 224\)](#)
- [Available Libraries \(p. 225\)](#)

Amazon CloudWatch Endpoints

For information about the regions and endpoints used with CloudWatch, see [Regions and Endpoints](#) in the [Amazon Web Services General Reference](#).

Query Parameters

Each query request must include some common parameters to handle authentication and selection of an action. For more information, see [Common Query Parameters](#) in the [Amazon CloudWatch API Reference](#).

Note

Some API operations take lists of parameters. These lists are specified using the following notation: param.member.n. Values of n are integers starting from 1. All lists of parameters must follow this notation, including lists that contain only one parameter. For example, a Query parameter list looks like this:

```
&attribute.member.1=this  
&attribute.member.2=that
```

The RequestId

In every response from Amazon Web Services (AWS), you will find `ResponseMetadata`, which contains a string element called `RequestId`. This is simply a unique identifier that AWS assigns to provide tracking information. Although `RequestId` is included as part of every response, it will not be listed on the individual API documentation pages to improve readability of the API documentation and to reduce redundancy.

Query API Authentication

You can send query requests over either HTTP or HTTPS. Regardless of which protocol you use, you must include a signature in every query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 2*. For more information about creating and including a signature, see [Signing AWS API Requests](#) in the *AWS General Reference*.

To create the signature

1. Create the canonicalized query string that you need later in this procedure:
 - a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when `Content-Type` is `application/x-www-form-urlencoded`).
 - b. URL encode the parameter name and values according to the following rules:
 - Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
 - Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
 - Percent encode extended UTF-8 characters in the form %XY%ZA....
 - Percent encode the space character as %20 (and not +, as common encoding schemes do).

Note

Currently all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

- c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
 - d. Separate the name-value pairs with an ampersand (&) (ASCII character 38).
-
2. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline character).

```
StringToSign = HTTPVerb + "\n" +
    ValueOfHostHeaderInLowercase + "\n" +
    HTTPRequestURI + "\n" +
    CanonicalizedQueryString <from the preceding step>
```

The `HTTPRequestURI` component is the HTTP absolute path component of the URI up to, but not including, the query string. If the `HTTPRequestURI` is empty, use a forward slash (/).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm.
For more information, see <http://www.ietf.org/rfc/rfc2104.txt>.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the *Signature* request parameter.

Important

The final signature you send in the request must be URL encoded as specified in RFC 3986 (for more information, see <http://www.ietf.org/rfc/rfc3986.txt>). If your toolkit URL encodes your final request, then it handles the required URL encoding of the signature. If your toolkit doesn't URL encode the final request, then make sure to URL encode the signature before you include it in the request. Most importantly, make sure the signature is URL encoded *only once*. A common mistake is to URL encode it manually during signature formation, and then again when the toolkit URL encodes the entire request.

Query API Examples Using Signature Version 2

Example ListMetrics API Request

This example uses the Amazon CloudWatch [ListMetrics](#) action.

```
http://monitoring.amazonaws.com/?SignatureVersion=2
&Action=ListMetrics
&Version=2010-08-01
&AWSAccessKeyId=<Your AWS Access Key Id>
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-11-17T05%3A13%3A00.000Z
```

The following is the string to sign.

```
GET\n
monitoring.amazonaws.com\n
/\n
AWSAccessKeyId=<Your AWS Access Key Id>
&Action=ListMetrics
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2010-11-17T05%3A13%3A00.000Z
&Version=2010-08-01
```

The following is the signed request.

```
http://monitoring.amazonaws.com/?Action=ListMetrics
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-11-17T05%3A13%3A00.000Z
&Signature=<URLEncode(Base64Encode(Signature))>
&Version=2010-08-01
&AWSAccessKeyId=<Your AWS Access Key Id>
```

Example ListMetrics API Request Using NextToken Value

This example uses the Amazon CloudWatch [ListMetrics](#) action and the `NextToken` value to retrieve more than 500 metrics.

1. Create the string to sign according to the following pseudo-grammar (the "`\n`" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

The `HTTPRequestURI` component is the HTTP absolute path component of the URI up to but not including the query string. If the `HTTPRequestURI` is empty, use a forward slash (/).

2. Prepare a string to sign, as in the following example:

```
GET\n
monitoring.amazonaws.com\n
/\n
AWSAccessKeyId=<Your AWS Access Key Id>
&Action=ListMetrics
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2010-11-17T05%3A13%3A00.000Z
&Version=2010-08-01
```

The following are specified in the example:

`Action` specifies the action to take (e.g., `ListMetrics`).

`AWSAccessKeyId` specifies the `AccessKeyId` for your AWS account (replace `<Your AWS Access Key ID>` with your actual `AWSAccessKeyId`).

`SignatureVersion` specifies the version of the signature (e.g., 2).

`Timestamp` 2012-09-27T17:06:23.000Z specifies the time stamp of the request (e.g., 09/27/2012, 17:06:23).

`Version` 2010-08-01 specifies the version of the `ListMetrics` API (e.g., last released on 2010-08-01).

3. Sign the request URL with the string from the previous step. The following example shows the signed request URL.

```
http://monitoring.amazonaws.com?SignatureVersion=2
&Action=ListMetrics
&Version=2010-08-01
&Timestamp=2012-09-27T17%3A14%3A01.000Z
&AWSAccessKeyId=<Your AWS Access Key ID>
&Signature=iE68300Pbl%2BDsKM5mFiOhHWEXAMPLE
```

The following are specified in the example:

`Action` specifies the action to take (e.g., `ListMetrics`).

`Version=2010-08-01` specifies the version of the ListMetrics API (e.g., last released on 2010-08-01).

`Timestamp=2012-09-27T17:06:23.000Z` specifies the time stamp of the request (e.g., 09/27/2012, 17:06:23).

`AWSAccessKeyId` specifies the AccessKeyId for your AWS account (replace <Your AWS Access Key ID> with your actual AWSAccessKeyId).

`Signature` specifies the signature for signing the request (e.g., <URLEncode(Base64Encode(Signature))>=iE68300Pbl%2BDsKM5mFiOhHWEXAMPLE).

4. Copy the signed request URL to your web browser and press Enter to run the request. You should get results similar to the following:

```
<ListMetricsResponse>
  <ListMetricsResult>
    <Metrics>
      <member>
        <Dimensions>
          <member>
            <Name>InstanceId</Name>
            <Value>i-8dea01f0</Value>
          </member>
        </Dimensions>
        <MetricName>CPUUtilization</MetricName>
        <Namespace>AWS/EC2<Namespace>
      </member>
      <member>
        <Dimensions>
          <member>
            <Name>InstanceId</Name>
            <Value>i-7dee09t3</Value>
          </member>
        </Dimensions>
        <MetricName>CPUUtilization</MetricName>
        <Namespace>AWS/EC2<Namespace>
      </member>
    </Metrics>
    <NextToken>NNNTTT</NextToken>
  </ListMetricsResult>
  <ResponseMetadata>
    <RequestId>8f95da07-08c8-11e2-9cdd-4d93ea583888</RequestId>
  </ResponseMetadata>
</ListMetricsResponse>
```

5. From the output in Step 4, save the `NextToken` value. In this example it is `NNNTTT`, but it is normally a very long string.
6. Prepare the string to sign and include the `NextToken=NNNTTT` value.
7. Sign the request URL with the string from Step 6. The following example is the signed request URL. It is similar to the signed request URL in Step 3, except that it has the `NextToken=NNNTTT` value.

```
http://monitoring.amazonaws.com?SignatureVersion=2
&Action=ListMetrics
&Version=2010-08-01
&NextToken=NNNTTT
```

```
&Timestamp=2012-09-27T17%3A45%3A14.000Z  
&AWSAccessKeyId=<Your AWS Access Key ID>  
&Signature=iE68300Pbl%2BDsKM5mFiOhHWEXAMPLE
```

8. To get the next 500 metrics, copy the signed request URL to your web browser, and press Enter. You can repeat the request with the same parameters and the new `NextToken` until you have retrieved all of the metrics.

Query API Error Messages Using Signature Version 2

Example Error Message When Using the Wrong AWS Secret Access Key to Calculate the Signature

```
<ErrorResponse>
  <Error>
    <Type>Sender</Type>
    <Code>SignatureDoesNotMatch</Code>
    <Message>
      The request signature we calculated does not match the signature you
      provided. Check your AWS Secret Access Key and signing method. Consult
      the service
      documentation for details.
    </Message>
  </Error>
  <RequestId>a9c1a06e-0d80-11e2-ac80-9646d110ca61<RequestId>
</ErrorResponse>
```

Example Error Message When Using the Wrong AWS AccessKeyID

```
<ErrorResponse>
  <Error>
    <Type>Sender</Type>
    <Code>InvalidClientTokenId</Code>
    <Message>
      The security token included in the request is invalid
    </Message>
  </Error>
  <RequestId>5134a3b8-0d82-11e2-9b0a-db3eb46b3dbd<RequestId>
</ErrorResponse>
```

Example Error Message When Providing Incorrect Parameters

The following example shows how to make a request with the MetricName parameter "TestMetric".

The signed URL looks like this:

```
http://monitoring.amazonaws.com?SignatureVersion=2
&Action=ListMetrics
&Version=2010-08-01
&MetricName=TestMetric
&Timestamp=2012-09-27T17%3A14%3A01.000Z
&AWSAccessKeyId=<Your AWS Access Key ID>
&Signature=iE68300Pbl%2BDsKM5mFiOhHWEXAMPLE
```

When you try to retrieve more metrics with the NextToken, you have to provide the same parameters as in the previous request. If you provide the wrong parameters, provide more parameters, or provide fewer parameters (assume that the NextToken is correct), you will get the following error:

```
<ErrorResponse>
  <Error>
    <Type>Sender</Type>
    <Code>InvalidParameterValue</Code>
    <Message>Invalid nextToken</Message>
  </Error>
  <RequestId>cb1c4191-0e5d-11e2-9c15-6f306828daaa<RequestId>
</ErrorResponse>
```

Available Libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of the command-line tools and Query API. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. Libraries and resources are available for the following languages and platforms:

- [Android](#)
- [iOS](#)
- [Java](#)
- [PHP](#)
- [Python](#)
- [Ruby](#)
- [Windows and .NET](#)

For libraries and sample code in all languages, see [Sample Code & Libraries](#).

Document History

The following table describes the important changes to the Amazon CloudWatch Developer Guide. This documentation is associated with the 2010-08-01 release of CloudWatch. This guide was last updated on 10 November 2014.

Change	Description	Release Date
Added support for AWS CloudTrail logged events in Amazon CloudWatch Logs	You can create alarms in CloudWatch and receive notifications of particular API activity as captured by CloudTrail and use the notification to perform troubleshooting. For more information, see Monitoring Log Files (p. 113) .	November 10, 2014
Added support for Amazon CloudWatch Logs	You can use Amazon CloudWatch Logs to monitor, store, and access your system, application, and custom log files from Amazon Elastic Compute Cloud (Amazon EC2) instances or other sources. You can then retrieve the associated log data from CloudWatch Logs using the Amazon CloudWatch console, the CloudWatch Logs commands in the AWS CLI, or the CloudWatch Logs SDK. For more information, see Monitoring Log Files (p. 113) .	July 10, 2014
Added Amazon Simple Workflow Service metrics and dimensions	Added Amazon Simple Workflow Service metrics and dimensions. For more information, see Amazon SWF Dimensions and Metrics (p. 213) .	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 in AWS CloudTrail (p. 147) .	30 April 2014

Change	Description	Release Date
Updated guide to use the new AWS command line interface (CLI)	<p>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.</p> <p>For information about how to install and configure the new AWS CLI, see Getting Set Up with the AWS Command Line Interface in the <i>AWS Command Line Interface User Guide</i>.</p>	21 February 2014
Updated guide to match new Amazon CloudWatch console	Updated topics to cover the brand new Amazon CloudWatch console.	29 October 2013
Rewrote Amazon CloudWatch Developer's Guide	Rewrote and restructured the Developer's Guide to include the Getting Started Guide, Getting Setup section, updated command line interface reference, and updated/additional procedures for working with metrics and alarms.	6 September 2013
Added Amazon Redshift and AWS OpsWorks metrics and dimensions	Added Amazon Redshift and AWS OpsWorks metrics and dimensions. For more information, see Amazon Redshift Dimensions and Metrics (p. 205) and AWS OpsWorks Dimensions and Metrics (p. 203) .	16 July 2013
Added Amazon Route 53 metrics and dimensions	Added Amazon Route 53 metrics and dimensions. For more information, see Amazon Route 53 Dimensions and Metrics (p. 210) .	26 June 2013
Updates to Amazon CloudWatch Monitoring Scripts for Linux	Added updates to the monitoring scripts for Linux to add support for AWS Identity and Access Management (IAM) roles, using reported metrics with Auto Scaling, and options for aggregated CloudWatch metrics. For more information, see Amazon CloudWatch Monitoring Scripts for Linux (p. 153) .	22 February 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 That Stop or Terminate an Instance (p. 88) .	8 January 2013
Updated EBS metrics	Updated the EBS metrics to include two new metrics for Provisioned IOPS volumes. For more information, see Amazon EBS Dimensions and Metrics (p. 191) .	20 November 2012
New scripts	You can now use the Amazon CloudWatch Monitoring Scripts for Windows to produce and consume Amazon CloudWatch custom metrics. For more information, see Amazon CloudWatch Monitoring Scripts for Windows (p. 160) .	19 July 2012

Change	Description	Release Date
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 Monitor Your Estimated Charges Using Amazon CloudWatch (p. 103) .	10 May 2012
New scripts	You can now use the Amazon CloudWatch Monitoring Scripts for Linux to produce and consume Amazon CloudWatch custom metrics. For more information, see Amazon CloudWatch Monitoring Scripts for Linux (p. 153) .	24 February 2012
New metrics	You can now access six new Elastic Load Balancing metrics that provide counts of various HTTP response codes. For more information, see Elastic Load Balancing Dimensions and Metrics (p. 196) .	19 October 2011
New feature	You can now access metrics from Amazon Elastic MapReduce. For more information, see the section called “Amazon Elastic MapReduce Dimensions and Metrics” (p. 198) .	30 June 2011
New feature	You can now access metrics from Amazon Simple Notification Service and Amazon Simple Queue Service. For more information, see the section called “Amazon Simple Notification Service Dimensions and Metrics” (p. 210) and the section called “Amazon SQS Dimensions and Metrics” (p. 211) .	14 July 2011
Restructured Guide	Renamed, merged, and moved sections, including the entire User Scenario section: <ul style="list-style-type: none"> • <i>CloudWatch Support for AWS Products</i> is now Amazon CloudWatch Namespaces, Dimensions, and Metrics Reference (p. 175) • <i>List Available Metrics</i> is now the section called “View Available Metrics” (p. 26) • <i>Get Statistics on a Metric</i> is now the section called “Get Statistics for a Metric” (p. 34) • <i>Create an Alarm that Sends Email</i> is now Creating Amazon CloudWatch Alarms (p. 71) 	01 July 2011
New section	Added a section that describes how to use AWS Identity and Access Management (IAM). For more information, see Controlling User Access to Your AWS Account (p. 170) .	7 June 2011
New Feature	Added information about using the PutMetricData API to publish custom metrics. For more information, see the section called “Publish Custom Metrics” (p. 68) or go to the Amazon CloudWatch Getting Started Guide .	10 May 2011

Change	Description	Release Date
Updated Content	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 link	This service's endpoint information is now located in the Amazon Web Services General Reference. For more information, go to Regions and Endpoints in Amazon Web Services General Reference .	2 March 2011
Updated Content	Added information about using the AWS Management Console to manage CloudWatch.	11 February 2011
Updated Content	Added a brief discussion about alarms and Auto Scaling. Specifically, alarms continue to invoke Auto Scaling policy notifications for the duration of a threshold breach rather than only after the initial breach. For more information, see Alarms (p. 8) .	19 January 2011
Updated Content	Removed Minimum and Maximum from the list of valid statistics for the Elastic Load Balancing RequestCount metric. The only valid statistic for RequestCount is Sum. For a list of all Elastic Load Balancing metrics, see Elastic Load Balancing Dimensions and Metrics (p. 196) .	18 January 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. 8) .	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). For more information, go to http://aws.amazon.com/iam and to the Using IAM .	02 December 2010

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.