
Amazon CloudWatch

User Guide



Amazon CloudWatch: User Guide

Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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
Accessing CloudWatch	1
Related AWS Services	1
How CloudWatch Works	2
Concepts	2
Namespaces	3
Metrics	3
Dimensions	4
Statistics	5
Percentiles	7
Alarms	7
Resources	8
Getting Set Up	9
Sign Up for Amazon Web Services (AWS)	9
Sign in to the Amazon CloudWatch Console	9
Set Up the AWS CLI	9
Getting Started	10
See Key Metrics From All AWS Services	12
Remove a Service from Appearing in the Cross Service Dashboard	13
Focus on a Single Service	14
Focus on a Resource Group	15
Using Dashboards	16
Create a Dashboard	17
Cross-Account Cross-Region Dashboards	17
Creating and Using a Cross-Account Cross-Region Dashboard With the AWS Management Console	18
Create a Cross-Account Cross-Region Dashboard Programmatically	19
Add or Remove a Graph	20
Move or Resize a Graph	22
Edit a Graph	23
Graph Metrics Manually on a CloudWatch Dashboard	24
Rename a Graph	25
Add or Remove a Text Widget	26
Add or Remove an Alarm	26
Use Live Data	27
Link and Unlink Graphs	28
Add a Dashboard to Your Favorites List	28
Change the Period Override Setting or Refresh Interval	28
Change the Time Range or Time Zone Format	29
Using Metrics	31
Viewing Available Metrics	31
Searching for Available Metrics	34
Getting Statistics for a Metric	35
Getting Statistics for a Specific Resource	35
Aggregating Statistics Across Resources	38
Aggregating Statistics by Auto Scaling Group	40
Aggregating Statistics by AMI	41
Graphing Metrics	42
Graphing a Metric	43
Using Dynamic Labels	46
Modifying the Time Range or Time Zone Format for a Graph	47
Modifying the Y-Axis for a Graph	48
Creating an Alarm from a Metric on a Graph	49
Publishing Custom Metrics	50

High-Resolution Metrics	50
Using Dimensions	50
Publishing Single Data Points	51
Publishing Statistic Sets	52
Publishing the Value Zero	52
Using Metric Math	52
Adding a Math Expression to a CloudWatch Graph	53
Metric Math Syntax and Functions	53
Using IF Expressions	61
Using Search Expressions in Graphs	63
Search Expression Syntax	64
Search Expression Examples	68
Creating a Graph with a Search Expression	69
Using Alarms	72
Metric Alarm States	72
Evaluating an Alarm	72
Configuring How Alarms Treat Missing Data	73
How Alarm State Is Evaluated When Data Is Missing	74
High-Resolution Alarms	76
Alarms on Math Expressions	76
Percentile-Based Alarms and Low Data Samples	76
Common Features of CloudWatch Alarms	76
Setting Up an SNS Topic	77
Setting Up an Amazon SNS Topic Using the AWS Management Console	77
Setting Up an SNS Topic Using the AWS CLI	78
Create an Alarm Based on a Static Threshold	79
Creating an Alarm Based on Anomaly Detection	82
Modifying an Anomaly Detection Model	83
Deleting an Anomaly Detection Model	84
Creating an Alarm Based on a Metric Math Expression	84
Creating a Composite Alarm	87
Editing or Deleting a CloudWatch Alarm	89
Creating a CPU Usage Alarm	90
Setting Up a CPU Usage Alarm Using the AWS Management Console	90
Setting Up a CPU Usage Alarm Using the AWS CLI	91
Creating a Load Balancer Latency Alarm	92
Setting Up a Latency Alarm Using the AWS Management Console	92
Setting Up a Latency Alarm Using the AWS CLI	93
Creating a Storage Throughput Alarm	93
Setting Up a Storage Throughput Alarm Using the AWS Management Console	93
Setting Up a Storage Throughput Alarm Using the AWS CLI	94
Create Alarms to Stop, Terminate, Reboot, or Recover an Instance	95
Adding Stop Actions to Amazon CloudWatch Alarms	96
Adding Terminate Actions to Amazon CloudWatch Alarms	97
Adding Reboot Actions to Amazon CloudWatch Alarms	97
Adding Recover Actions to Amazon CloudWatch Alarms	98
Viewing the History of Triggered Alarms and Actions	100
Creating a Billing Alarm	100
Enabling Billing Alerts	100
Creating a Billing Alarm	101
Deleting a Billing Alarm	102
Hiding Amazon EC2 Auto Scaling Alarms	102
Using Synthetic Monitoring	103
Required Roles and Permissions	104
Creating a Canary	107
Resources That Are Created for Canaries	108
Using Canary Blueprints	108

Canary Runtime Versions	111
Writing a Canary Script	112
Library Functions Available for Canary Scripts	115
Running a Canary on a VPC	120
Viewing Canary Statistics and Details	121
Deleting a Canary	122
Using ServiceLens to Monitor the Health of Your Applications	123
Deploying ServiceLens	124
Deploying AWS X-Ray	124
Deploying the CloudWatch Agent and the X-Ray Daemon	126
Using the Service Map	132
Using the Traces View	134
ServiceLens Troubleshooting	134
I Don't See All My Logs	134
I Don't See All My Alarms on the Service Map	135
I Don't See Some AWS Resources on the Service Map	135
There Are Too Many Nodes on My Service Map	136
Cross-Account Cross-Region CloudWatch Console	137
Enabling Cross-Account Cross-Region Functionality	137
(Optional) Integrate With AWS Organizations	139
Troubleshooting	140
Using Anomaly Detection	141
How Anomaly Detection Works	143
Using Contributor Insights to Analyze High-Cardinality Data	144
Creating a Contributor Insights Rule	144
Contributor Insights Rule Syntax	147
Example Rules	149
Viewing Contributor Insights Reports	151
Graphing Metrics Generated by Rules	152
Setting an Alarm on Contributor Insights Metric Data	152
Using Contributor Insights Built-In Rules	154
Using Container Insights	155
Setting Up Container Insights	157
Setting Up Container Insights on Amazon ECS	157
Setting Up Container Insights on Amazon EKS and Kubernetes	164
Viewing Container Insights Metrics	176
Using CloudWatch Logs Insights to View Container Insights Data	176
Metrics Collected by Container Insights	178
Amazon ECS Container Insights Metrics	178
Amazon EKS and Kubernetes Container Insights Metrics	182
Performance Log Reference	185
Performance Log Events for Amazon ECS	185
Performance Log Events for Amazon EKS and Kubernetes	188
Relevant Fields in Performance Log Events for Amazon EKS and Kubernetes	199
Container Insights Prometheus Metrics Monitoring	205
Install the CloudWatch Agent with Prometheus Metrics Collection	206
Configuring the CloudWatch Agent for Prometheus Monitoring	208
(Optional) Set Up Sample Containerized Workloads for Prometheus Metric Testing	215
Prometheus Metrics Collected by the CloudWatch Agent	224
Viewing Your Prometheus Metrics	229
Prometheus Metrics Troubleshooting	230
Troubleshooting Container Insights	233
Failed Deployment on Amazon EKS or Kubernetes	233
Unauthorized panic: Cannot retrieve cadvisor data from kubelet	233
Deploying Container Insights on a Deleted and Re-Created Cluster	233
Invalid Endpoint Error	234
Metrics Don't Appear in the Console	234

CrashLoopBackoff Error on the CloudWatch Agent	234
CloudWatch Agent or FluentD Pod Stuck in Pending	234
Building Your Own CloudWatch Agent Docker Image	234
Deploying Other CloudWatch Agent Features in Your Containers	234
Collect Metrics and Logs with the CloudWatch Agent	236
Installing the CloudWatch Agent	237
Installing the CloudWatch Agent Using the Command Line	237
Install the CloudWatch Agent Using Systems Manager	248
Installing the CloudWatch Agent Using AWS CloudFormation	261
Verifying the Signature of the CloudWatch Agent Package	265
Create the CloudWatch Agent Configuration File	270
Create the CloudWatch Agent Configuration File with the Wizard	270
Manually Create or Edit the CloudWatch Agent Configuration File	275
Metrics Collected by the CloudWatch Agent	305
Metrics Collected by the CloudWatch Agent on Windows Server Instances	305
Metrics Collected by the CloudWatch Agent on Linux Instances	305
Common Scenarios with the CloudWatch Agent	313
Running the CloudWatch Agent as a Different User	313
Adding Custom Dimensions to Metrics Collected by the CloudWatch Agent	315
Multiple CloudWatch Agent Configuration Files	315
Aggregating or Rolling Up Metrics Collected by the CloudWatch Agent	317
Collecting High-Resolution Metrics With the CloudWatch agent	317
Sending Metrics and Logs to a Different Account	318
Timestamp Differences Between the Unified CloudWatch Agent and the Older CloudWatch Logs Agent	320
Troubleshooting the CloudWatch Agent	320
CloudWatch Agent Command Line Parameters	320
Installing the CloudWatch Agent Using Run Command Fails	321
The CloudWatch Agent Won't Start	321
Verify That the CloudWatch Agent Is Running	321
Where Are the Metrics?	322
The CloudWatch Agent Won't Start, and the Error Mentions an Amazon EC2 Region	322
The CloudWatch Agent Won't Start on Windows Server	322
Unable to Find Credentials on Windows Server	323
CloudWatch Agent Files and Locations	323
Logs Generated by the CloudWatch Agent	324
Stopping and Restarting the CloudWatch Agent	324
Detect common application problems with CloudWatch Application Insights for .NET and SQL Server	326
What Is CloudWatch Application Insights?	326
Features	327
Concepts	327
Pricing	328
Related AWS services	328
Supported application components	329
Supported technology stack	329
How CloudWatch Application Insights works	329
How Application Insights monitors applications	330
Data retention	330
Quotas	330
Getting started	331
Accessing CloudWatch Application Insights for .NET and SQL Server	331
Prerequisites	331
IAM policy	332
Setting up your application	333
Component configuration	344
JSON	344
Component configuration sections	344

View and troubleshoot Application Insights	353
Configuration errors	354
Logs and metrics	355
Amazon Elastic Compute Cloud (EC2)	356
Elastic Load Balancer (ELB)	362
Application ELB	362
Amazon EC2 Auto Scaling Groups	362
Amazon Simple Queue Server (SQS)	363
Amazon Relational Database Service (RDS)	363
AWS Lambda Function	364
Amazon DynamoDB table	364
Amazon S3 bucket	365
Metrics With datapoints requirements	365
Recommended metrics	370
Performance Counter metrics	374
Services That Publish CloudWatch Metrics	378
Alarm Events and EventBridge	383
Sample Events from CloudWatch	383
Ingesting High-Cardinality Logs and Generating Metrics with CloudWatch Embedded Metric Format	387
Generating Logs Using the Embedded Metric Format	387
Using the Client Libraries	388
Manually Generating Embedded Metric Format Logs	388
Viewing Your Metrics and Logs in the Console	400
Monitor Applications Using AWS SDK Metrics	402
Metrics and Data Collected by SDK Metrics for Enterprise Support	403
Set Up SDK Metrics	405
Configure the CloudWatch Agent for SDK Metrics	406
Set IAM Permissions for SDK Metrics	407
Service Quotas Integration and Usage Metrics	409
Visualizing Your Service Quotas and Setting Alarms	409
CloudWatch Usage Metrics	410
CloudWatch Tutorials	412
Scenario: Monitor Estimated Charges	412
Step 1: Enable Billing Alerts	412
Step 2: Create a Billing Alarm	413
Step 3: Check the Alarm Status	413
Step 4: Edit a Billing Alarm	414
Step 5: Delete a Billing Alarm	414
Scenario: Publish Metrics	414
Step 1: Define the Data Configuration	415
Step 2: Add Metrics to CloudWatch	415
Step 3: Get Statistics from CloudWatch	416
Step 4: View Graphs with the Console	416
Tagging Your CloudWatch Resources	417
Supported Resources in CloudWatch	417
Managing Tags	417
Tag Naming and Usage Conventions	418
Security	419
Data Protection	419
Encryption in Transit	420
Identity and Access Management	420
Authentication	420
Access Control	421
CloudWatch Dashboard Permissions Update	421
Overview of Managing Access	422
Using Identity-Based Policies (IAM Policies)	425
Using Condition Keys to Limit Access to CloudWatch Namespaces	431

Using Service-Linked Roles	432
Using Service-Linked Roles for Application Insights	436
Amazon CloudWatch Permissions Reference	439
Compliance Validation	446
Resilience	446
Infrastructure Security	446
Network Isolation	447
Interface VPC Endpoints	447
CloudWatch	447
CloudWatch Synthetics	449
Security Considerations for Synthetics Canaries	450
Use Secure Connections	450
Canary Naming Considerations	451
Secrets in Canary Code	451
Permissions Considerations	451
Stack Traces and Exception Messages	451
Scope Your IAM Roles Narrowly	451
Header Logging	452
Logging API Calls with AWS CloudTrail	453
CloudWatch Information in CloudTrail	453
Example: CloudWatch Log File Entries	454
CloudWatch Synthetics Information in CloudTrail	456
Example: CloudWatch Synthetics Log File Entries	456
Grafana Integration	459
Service Quotas	460
Document History	463

What Is Amazon CloudWatch?

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

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

You can create alarms which watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use this data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money.

With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.

Accessing CloudWatch

You can access CloudWatch using any of the following methods:

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

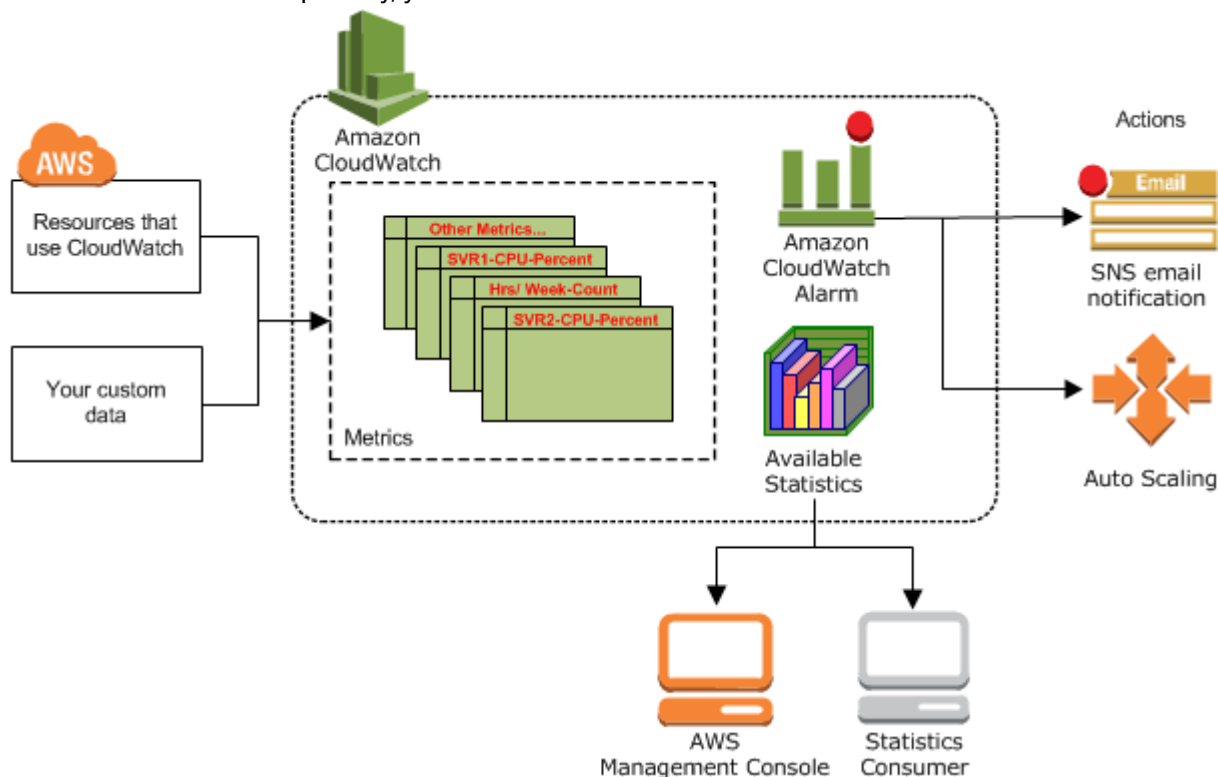
Related AWS Services

The following services are used along with Amazon CloudWatch:

- **Amazon Simple Notification Service (Amazon SNS)** coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. You use Amazon SNS with CloudWatch to send messages when an alarm threshold has been reached. For more information, see [Setting Up Amazon SNS Notifications \(p. 77\)](#).
- **Amazon EC2 Auto Scaling** enables you to automatically launch or terminate Amazon EC2 instances based on user-defined policies, health status checks, and schedules. You can use a CloudWatch alarm with Amazon EC2 Auto Scaling to scale your EC2 instances based on demand. For more information, see [Dynamic Scaling](#) in the *Amazon EC2 Auto Scaling User Guide*.
- **AWS CloudTrail** enables you to monitor the calls made to the Amazon CloudWatch API for your account, including calls made by the AWS Management Console, AWS CLI, and other services. When CloudTrail logging is turned on, CloudWatch writes log files to the Amazon S3 bucket that you specified when you configured CloudTrail. For more information, see [Logging Amazon CloudWatch API Calls with AWS CloudTrail \(p. 453\)](#).
- **AWS Identity and Access Management (IAM)** is a web service that helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication) and what resources they can use in which ways (authorization). For more information, see [Identity and Access Management for Amazon CloudWatch \(p. 420\)](#).

How Amazon CloudWatch Works

Amazon CloudWatch is basically a metrics repository. An AWS service—such as Amazon EC2—puts metrics into the repository, and you retrieve statistics based on those metrics. If you put your own custom metrics into the repository, you can retrieve statistics on these metrics as well.



You can use metrics to calculate statistics and then present the data graphically in the CloudWatch console. For more information about the other AWS resources that generate and send metrics to CloudWatch, see [AWS Services That Publish CloudWatch Metrics \(p. 378\)](#).

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

AWS Cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, each data center facility is located in a specific geographical area, known as a *Region*. Each Region is designed to be completely isolated from the other Regions, to achieve the greatest possible failure isolation and stability. Metrics are stored separately in Regions, but you can use CloudWatch cross-Region functionality to aggregate statistics from different Regions. For more information, see [Cross-Account Cross-Region CloudWatch Console \(p. 137\)](#) and [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Amazon CloudWatch Concepts

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

- [Namespaces \(p. 3\)](#)

- [Metrics \(p. 3\)](#)
- [Dimensions \(p. 4\)](#)
- [Statistics \(p. 5\)](#)
- [Percentiles \(p. 7\)](#)
- [Alarms \(p. 7\)](#)

Namespaces

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

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

The AWS namespaces typically use the following naming convention: `AWS/service`. For example, Amazon EC2 uses the `AWS/EC2` namespace. For the list of AWS namespaces, see [AWS Services That Publish CloudWatch Metrics \(p. 378\)](#).

Metrics

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

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

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

Metrics are uniquely defined by a name, a namespace, and zero or more dimensions. Each data point in a metric has a time stamp, and (optionally) a unit of measure. You can retrieve statistics from CloudWatch for any metric.

For more information, see [Viewing Available Metrics \(p. 31\)](#) and [Publishing Custom Metrics \(p. 50\)](#).

Time Stamps

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

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

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

Metrics Retention

CloudWatch retains metric data as follows:

- Data points with a period of less than 60 seconds are available for 3 hours. These data points are high-resolution custom metrics.
- Data points with a period of 60 seconds (1 minute) are available for 15 days
- Data points with a period of 300 seconds (5 minute) are available for 63 days
- Data points with a period of 3600 seconds (1 hour) are available for 455 days (15 months)

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

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

Dimensions

A *dimension* is a name/value pair that is part of the identity of a metric. You can assign up to 10 dimensions to a metric.

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

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

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

Dimension Combinations

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

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

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

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

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

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

You can't retrieve statistics for the following dimensions or if you specify no dimensions. (The exception is by using the metric math **SEARCH** function, which can retrieve statistics for multiple metrics. For more information, see [Using Search Expressions in Graphs \(p. 63\)](#).)

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

Statistics

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

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

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

Units

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

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

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

Periods

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

Only custom metrics that you define with a storage resolution of 1 second support sub-minute periods. Even though the option to set a period below 60 is always available in the console, you should select a period that aligns to how the metric is stored. For more information about metrics that support sub-minute periods, see [High-Resolution Metrics](#) (p. 50).

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

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

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

Aggregation

Amazon CloudWatch aggregates statistics according to the period length that you specify when retrieving statistics. You can publish as many data points as you want with the same or similar time stamps. CloudWatch aggregates them according to the specified period length. CloudWatch does not aggregate data across Regions.

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

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

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

Percentiles

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

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

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

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

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

- The SampleCount value of the statistic set is 1 and Min, Max, and Sum are all equal.
- The Min and Max are equal, and Sum is equal to Min multiplied by SampleCount.

Alarms

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

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

When creating an alarm, select a period that is greater than or equal to the frequency of the metric to be monitored. For example, basic monitoring for Amazon EC2 provides metrics for your instances every 5 minutes. When setting an alarm on a basic monitoring metric, select a period of at least 300 seconds (5 minutes). Detailed monitoring for Amazon EC2 provides metrics for your instances every 1 minute. When setting an alarm on a detailed monitoring metric, select a period of at least 60 seconds (1 minute).

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. There is a higher charge for high-resolution alarms. For more information about high-resolution metrics, see [Publishing Custom Metrics \(p. 50\)](#).

For more information, see [Using Amazon CloudWatch Alarms \(p. 72\)](#) and [Creating an Alarm from a Metric on a Graph \(p. 49\)](#).

Amazon CloudWatch Resources

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

Resource	Description
Amazon CloudWatch FAQs	The FAQ covers the top questions developers have asked about this product.
Release notes	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code examples, release notes, and other information to help you build innovative applications with AWS.
AWS Management Console	The console allows you to perform most of the functions of Amazon CloudWatch and various other AWS offerings without programming.
Amazon CloudWatch Discussion Forums	Community-based forum for developers to discuss technical questions related to Amazon CloudWatch.
AWS Support	The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
Amazon CloudWatch product information	The primary webpage for information about Amazon CloudWatch.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse, etc.

Getting Set Up

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

Sign Up for Amazon Web Services (AWS)

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

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

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

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

Sign in to the Amazon CloudWatch Console

To sign in to the Amazon CloudWatch console

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

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

Set Up the AWS CLI

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

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

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

Getting Started with Amazon CloudWatch

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

The CloudWatch overview home page appears.

CloudWatch: Overview ▾

All resources ▾

AWS services summary ⓘ

Services

Status

Alarm

❗ EC2

1

❗ Lambda

2

❗ RDS

1

⚠ Kinesis

-

✅ DynamoDB

-

❓ API Gateway

-

❓ Billing

-

❓ Classic ELB

-

❓ CloudFront

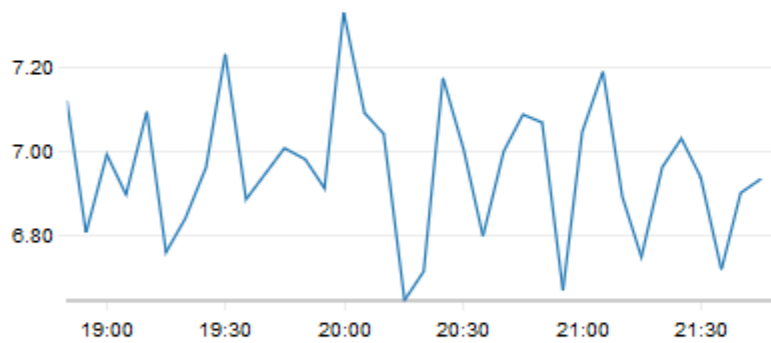
-

❓ CloudWatch Events

Default dashboard ⓘ [Edit dashboard](#)

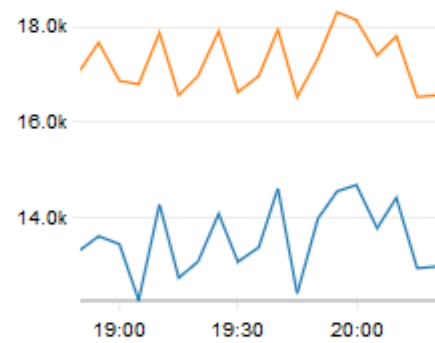
Custom metric 1

Percent



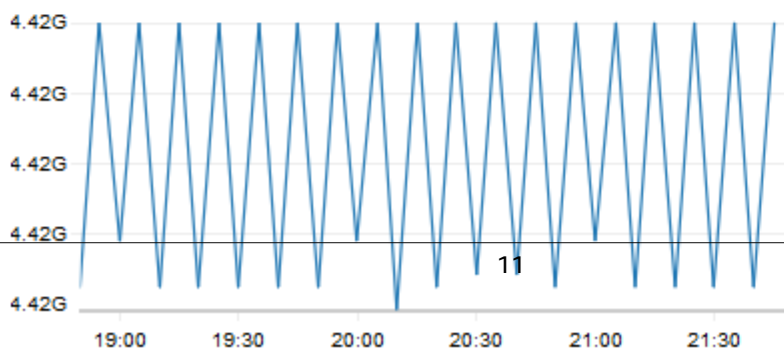
Custom metric 2

Bytes



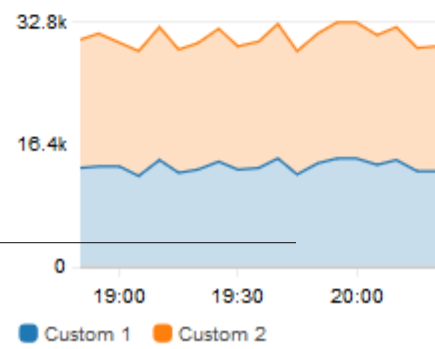
Custom metrics 5

Bytes



Custom metrics 2

Bytes



The overview displays the following items, refreshed automatically.

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

These upper areas enable you to assess the health of your AWS services, by seeing the alarm states in every service and the alarms that most recently changed state. This helps you monitor and quickly diagnose issues.

- Below these areas is the *default dashboard*, if one exists. The default dashboard is a custom dashboard that you have created and named **CloudWatch-Default**. This is a convenient way for you to add metrics about your own custom services or applications to the overview page, or to bring forward additional key metrics from AWS services that you most want to monitor.
- If you use six or more AWS services, below the default dashboard is a link to the automatic cross-service dashboard. The cross-service dashboard automatically displays key metrics from every AWS service you use, without requiring you to choose what metrics to monitor or create custom dashboards. You can also use it to drill down to any AWS service and see even more key metrics for that service.

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

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

Topics

- [See Key Metrics From All AWS Services \(p. 12\)](#)
- [Focus on Metrics and Alarms in a Single AWS Service \(p. 14\)](#)
- [Focus on Metrics and Alarms in a Resource Group \(p. 15\)](#)

See Key Metrics From All AWS Services

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

To open the cross service dashboard

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

The overview appears.

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

The cross-service dashboard appears, showing each AWS service you are using, displayed in alphabetical order. For each service, one or two key metrics are displayed.

3. You can focus on a particular service in two ways:
 - a. To see more key metrics for a service, choose its name from the list at the top of the screen, where **Cross service dashboard** is currently shown. Or, you can choose **View Service dashboard** next to the service name.

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

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

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

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

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

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

For more information, see [Focus on Metrics and Alarms in a Resource Group \(p. 15\)](#).

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

Remove a Service from Appearing in the Cross Service Dashboard

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

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

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

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

The home page appears.

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

The view changes to show metrics from only that service.

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

Focus on Metrics and Alarms in a Single AWS Service

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

To focus on a single service

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

The home page appears.

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

The view changes to display graphs of key metrics from the selected service.

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

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

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

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

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

The CloudWatch Logs view appears in a new tab, displaying a list of your log groups. To see the log events in one of these log groups that occurred during the time range shown in the original graph, choose that log group.

5. When viewing alarms, you can focus on a particular alarm in several ways:

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

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

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

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

Focus on Metrics and Alarms in a Resource Group

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

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

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

To focus on a single resource group

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

The cross-service dashboard appears, showing only the services related to this resource group. For each service, one or two key metrics are displayed.

4. To change the time range shown in all graphs and alarms currently displayed, for **Time range** at the top of the screen, select a range. To select from more time range options than those displayed by default, choose **custom**.
5. Alarms are always refreshed one time per minute. To refresh the view, choose the refresh icon (two curved arrows) at the top right of the screen. To change the automatic refresh rate for items on the screen other than alarms, choose the down arrow next to the refresh icon and choose a refresh rate. You can also choose to turn off automatic refresh.
6. To return to showing information about all the resources in your account, near the top of the screen where the name of the resource group is currently displayed, choose **All resources**.

Using Amazon CloudWatch Dashboards

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

With dashboards, you can create the following:

- A single view for selected metrics and alarms to help you assess the health of your resources and applications across one or more regions. You can select the color used for each metric on each graph, so that you can easily track the same metric across multiple graphs.

You can create dashboards that display graphs and other widgets from multiple AWS accounts and multiple Regions. For more information, see [Cross-Account Cross-Region CloudWatch Console \(p. 137\)](#).

- An operational playbook that provides guidance for team members during operational events about how to respond to specific incidents.
- A common view of critical resource and application measurements that can be shared by team members for faster communication flow during operational events.

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

To access CloudWatch dashboards, you need one of the following:

- The `AdministratorAccess` policy
- The `CloudWatchFullAccess` policy
- A custom policy that includes one or more of these specific permissions:
 - `cloudwatch:GetDashboard` and `cloudwatch:ListDashboards` to be able to view dashboards
 - `cloudwatch:PutDashboard` to be able to create or modify dashboards
 - `cloudwatch:DeleteDashboards` to be able to delete dashboards

Contents

- [Creating a CloudWatch Dashboard \(p. 17\)](#)
- [Cross-Account Cross-Region Dashboards \(p. 17\)](#)
- [Add or Remove a Graph from a CloudWatch Dashboard \(p. 20\)](#)
- [Move or Resize a Graph on a CloudWatch Dashboard \(p. 22\)](#)
- [Edit a Graph on a CloudWatch Dashboard \(p. 23\)](#)
- [Graph Metrics Manually on a CloudWatch Dashboard \(p. 24\)](#)
- [Rename a Graph on a CloudWatch Dashboard \(p. 25\)](#)
- [Add or Remove a Text Widget from a CloudWatch Dashboard \(p. 26\)](#)
- [Add or Remove an Alarm from a CloudWatch Dashboard \(p. 26\)](#)
- [Use Live Data \(p. 27\)](#)
- [Link and Unlink Graphs on a CloudWatch Dashboard \(p. 28\)](#)

- [Add a Dashboard to Your Favorites List \(p. 28\)](#)
- [Change the Period Override Setting or Refresh Interval for the CloudWatch Dashboard \(p. 28\)](#)
- [Change the Time Range or Time Zone Format of a CloudWatch Dashboard \(p. 29\)](#)

Creating a CloudWatch Dashboard

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

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

For more information about creating a dashboard using the API, see [PutDashboard](#) in the Amazon CloudWatch API Reference.

To create a dashboard using the console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and then **Create dashboard**.
3. In the **Create new dashboard** dialog box, enter a name for the dashboard and choose **Create dashboard**.

If you use the name **CloudWatch-Default**, the dashboard appears on the overview on the CloudWatch home page. For more information, see [Getting Started with Amazon CloudWatch \(p. 10\)](#).

If you use resource groups and name the dashboard **CloudWatch-Default-ResourceGroupName**, it appears on the CloudWatch home page when you focus on that resource group.

4. Do one of the following in the **Add to this dashboard** dialog box:
 - To add a graph to your dashboard, choose **Line** or **Stacked area** and choose **Configure**. In the **Add metric graph** dialog box, select the metrics to graph and choose **Create widget**. If a specific metric doesn't appear in the dialog box because it hasn't published data in more than 14 days, you can add it manually. For more information, see [Graph Metrics Manually on a CloudWatch Dashboard \(p. 24\)](#).
 - To add a number displaying a metric to the dashboard, choose **Number** and then **Configure**. In the **Add metric graph** dialog box, select the metrics to graph and choose **Create widget**.
 - To add a text block to your dashboard, choose **Text** and then **Configure**. In the **New text widget** dialog box, for **Markdown**, add and format your text using [Markdown](#). Choose **Create widget**.
5. Optionally, choose **Add widget** and repeat step 4 to add another widget to the dashboard. You can repeat this step multiple times.
6. Choose **Save dashboard**.

Cross-Account Cross-Region Dashboards

You can create *cross-account cross-Region dashboards*, which summarize your CloudWatch data from multiple AWS accounts and multiple Regions into one dashboard. From this high-level dashboard you can get a view of your entire application, and also drill down into more specific dashboards without having to log in and out of accounts or switch Regions.

You can create cross-account cross-Region dashboards in the AWS Management Console and programmatically.

Pre-Requisite

Before you can create a cross-account cross-Region dashboard, you must enable at least one sharing account and at least one monitoring account. Additionally, to be able to use the AWS Management Console to create a cross-account dashboard, you must enable the your console for cross-account functionality. For more information, see [Cross-Account Cross-Region CloudWatch Console \(p. 137\)](#).

Creating and Using a Cross-Account Cross-Region Dashboard With the AWS Management Console

You can use the AWS Management Console to create a cross-account cross-Region dashboard.

To create your a cross-account cross-Region dashboard

1. Log in to the monitoring account.
2. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
3. In the navigation pane, choose **Dashboards**.
4. Choose a dashboard, or create a new one.
5. At the top of the screen, use the boxes by **View data for** to switch between accounts and Regions.

As you create your dashboard, you can include widgets from multiple accounts and Regions. Widgets include graphs, alarms, and CloudWatch Logs Insights widgets.

Creating a graph with metrics from different accounts and Regions

1. Log in to the monitoring account.
2. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
3. In the navigation pane, choose **Metrics**.
4. Under **All metrics**, use the boxes by **View data for** to choose the account and Region that you want to add metrics from.
5. Add the metrics you want to the graph. For more information, see [Graphing Metrics \(p. 42\)](#).
6. Repeat steps 4-5 to add metrics from other accounts and Regions.
7. (Optional) Choose the **Graphed metrics** tab and add a metric math function that uses the metrics that you have chosen. For more information, see [Using Metric Math \(p. 52\)](#).

You can also set up a single graph to include multiple `SEARCH` functions. Each search can refer to a different account or Region.

8. When you are finished with the graph, choose **Actions, Add to dashboard**.

Select your cross-account dashboard, and choose **Add to dashboard**.

Adding an alarm from a different account to your cross-account dashboard

1. Log in to the monitoring account.
2. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
3. Use the boxes by **View data for** to choose the account where the alarm is located.
4. In the navigation pane, choose **Alarms**.
5. Select the check box next to the alarm that you want to add, and choose **Add to dashboard**.
6. Select the cross-account dashboard that you want to add it to, and choose **Add to dashboard**.

Create a Cross-Account Cross-Region Dashboard Programmatically

You can use the AWS APIs and SDKs to create dashboards programmatically. For more information, see [PutDashboard](#).

To enable cross-account cross-Region dashboards, we have added new parameters to the dashboard body structure, as shown in the following table and examples. For more information about overall dashboard body structure, see [Dashboard Body Structure and Syntax](#).

Parameter	Use	Scope	Default
accountId	Specifies the ID of the account where the widget or the metric is located.	Widget or metric	Account that is currently logged in
region	Specifies the Region of the metric.	Widget or metric	Current Region selected in the console

The following examples illustrate the JSON source for widgets in a cross-account cross-Region dashboard.

This example sets the `accountId` field to the ID of the sharing account at the widget level. This specifies that all metrics in this widget will come from that sharing account and Region.

```
{
  "widgets": [
    {
      ...
      "properties": {
        "metrics": [
          ...
        ],
        "accountId": "111122223333",
        "region": "us-east-1"
      }
    }
  ]
}
```

This example sets the `accountId` field differently at the level of each metric. In this example, the different metrics in this metric math expression come from different sharing accounts and different Regions.

```
{
  "widgets": [
    {
      ...
      "properties": {
        "metrics": [
          [ { "expression": "SUM(METRICS())", "label": "[avg: ${AVG}] Expression1", "id": "e1", "stat": "Sum" } ],
          [ "AWS/EC2", "CPUUtilization", { "id": "m2", "accountId": "5555666677778888", "region": "us-east-1", "label": "[avg: ${AVG}] ApplicationALabel " } ],
          [ ".", ".", { "id": "m1", "accountId": "9999000011112222", "region": "eu-west-1", "label": "[avg: ${AVG}] ApplicationBLabel " } ]
        ],
      }
    }
  ]
}
```

```
        "view": "timeSeries",
        "stacked": false,
        "stat": "Sum",
        "period": 300,
        "title": "Cross account example"
      }
    ]
  }
}
```

This example shows an alarm widget.

```
{
  "type": "metric",
  "x": 6,
  "y": 0,
  "width": 6,
  "height": 6,
  "properties": {
    "accountID": "111122223333",
    "title": "over50",
    "annotations": {
      "alarms": [
        "arn:aws:cloudwatch:us-east-1:379642911888:alarm:over50"
      ]
    },
    "view": "timeSeries",
    "stacked": false
  }
}
```

This example is for a CloudWatch Logs Insights widget.

```
{
  "type": "log",
  "x": 0,
  "y": 6,
  "width": 24,
  "height": 6,
  "properties": {
    "query": "SOURCE 'route53test' | fields @timestamp, @message\n| sort @timestamp\n| desc\n| limit 20",
    "accountId": "111122223333",
    "region": "us-east-1",
    "stacked": false,
    "view": "table"
  }
}
```

Another way to create dashboards programmatically is to first create one in the AWS Management Console, and then copy the JSON source of this dashboard. To do so, load the dashboard and choose **Actions, View/edit source**. You can then copy this dashboard JSON to use as a template to create similar dashboards.

Add or Remove a Graph from a CloudWatch Dashboard

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

To add a graph to a dashboard

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Choose **Add widget**.
4. Choose either **Line** or **Stacked area** and choose **Configure**.
5. In the **All metrics** tab, select the metrics to graph. If a specific metric doesn't appear in the dialog box because it hasn't published data in more than 14 days, you can add it manually. For more information, see [Graph Metrics Manually on a CloudWatch Dashboard \(p. 24\)](#).
6. (Optional) As you choose metrics to graph, you can specify a dynamic label to appear on the graph legend for each metric. Dynamic labels display a statistic about the metric and automatically update when the dashboard or graph is refreshed. To add a dynamic label, choose **Graphed metrics** and then **Dynamic labels**.

By default, the dynamic values you add to the label appear at the beginning of the label. You can then choose the **Label** value for the metric to edit the label. For more information, see [Using Dynamic Labels \(p. 46\)](#).

7. (Optional) As you choose metrics to graph, you can change their color on the graph. To do so, choose **Graphed metrics** and select the color square next to the metric to display a color picker box. Choose another color square in the color picker. Click outside the color picker to see your new color on the graph. Alternatively, in the color picker, you can enter the six-digit standard HTML hex color code for the color you want and press ENTER.
8. (Optional) To view more information about the metric being graphed, hover over the legend.
9. (Optional) To change the widget type, hover over the title area of the graph and choose **Widget actions, Widget type**.
10. (Optional) To change the statistic used for a metric, choose **Graphed metrics, Statistic**, and select the statistic you want to use. For more information, see [Statistics \(p. 5\)](#).
11. (Optional) To change the time range shown on the graph, choose either **custom** at the top of the graph or one of the time periods to the left of **custom**.
12. (Optional) Horizontal annotations help dashboard users quickly see when a metric has spiked to a certain level, or whether the metric is within a predefined range. To add a horizontal annotation, choose **Graph options, Add horizontal annotation**:
 - a. For **Label**, enter a label for the annotation.
 - b. For **Value**, enter the metric value where the horizontal annotation appears.
 - c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose **Above** or **Below** for the corresponding area to be filled. If you specify **Between**, another **Value** field appears, and the area of the graph between the two values is filled.
 - d. For **Axis**, specify whether the numbers in **Value** refer to the metric associated with the left y-axis or the right y-axis, if the graph includes multiple metrics.

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

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

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

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

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

- a. For **Label**, enter a label for the annotation. To show only the date and time on the annotation, keep the **Label** field blank.
- b. For **Date**, specify the date and time where the vertical annotation appears.
- c. For **Fill**, specify whether to use fill shading before or after a vertical annotation or between two vertical annotations. For example, choose **Before** or **After** for the corresponding area to be filled. If you specify **Between**, another **Date** field appears, and the area of the graph between the two values is filled.

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

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

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

14. Choose **Create widget**.
15. Choose **Save dashboard**.

To remove a graph from a dashboard

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

Move or Resize a Graph on a CloudWatch Dashboard

You can arrange and resize graphs on your CloudWatch dashboard.

To move a graph on a dashboard

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

To resize a graph

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

To enlarge a graph temporarily

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

2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Select the graph. Alternatively, hover over the title of the graph and choose **Widget actions**, **Enlarge**.

Edit a Graph on a CloudWatch Dashboard

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

To edit a graph on a dashboard

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Hover over the title of the graph and choose **Widget actions**, **Edit**.
4. To change the graph's title, select the title, enter a new title, and press ENTER.
5. To change the time range shown on the graph, choose either **custom** at the top of the graph, or one of the time periods to the left of **custom**.
6. To change the type of widget between separate lines on a graph, stacked lines on a graph, and a number, choose the box next to the right of **custom** and select either **Line**, **Stacked area**, or **Number**.
7. In the lower half of the screen, in the **Graphed metrics** tab, you can change the dynamic label, colors, statistic, or period corresponding to a metric:
 - a. (Optional) To specify a dynamic label for a metric, choose **Dynamic labels**. Dynamic labels display a statistic about the metric, and automatically update when the dashboard or graph is refreshed.

By default, the dynamic values you add to the label appear at the beginning of the label. You can then choose the **Label** value for the metric to edit the label. For more information, see [Using Dynamic Labels \(p. 46\)](#).
 - b. To change the color of one of the lines, select the color square next to the metric to display a color picker box. Choose another color in the color picker, and click outside the color picker to see your new color on the graph. Alternatively, in the color picker, you can enter the six-digit HTML hex color code for the color you want and press ENTER.
 - c. To change the statistic, choose **Statistic** in the lower half of the window, and choose the new statistic you want. For more information, see [Statistics \(p. 5\)](#).
 - d. To change the time period, which is next to **Statistic** in the lower half of the window, choose **Period** and select another value. This new setting is used on the dashboard only if the period setting of the dashboard itself is set to **Auto**. Otherwise, the period setting of the dashboard overrides the period setting for individual widgets.
8. To add or edit horizontal annotations, choose **Graph options**:
 - a. To add a horizontal annotation, choose **Add horizontal annotation**.
 - b. For **Label**, enter a label for the annotation.
 - c. For **Value**, enter the metric value where the horizontal annotation appears.
 - d. For **Fill**, specify how to use fill shading with this annotation. For example, choose **Above** or **Below** for the corresponding area to be filled. If you specify **Between**, another **Value** field appears, and the area of the graph between the two values is filled.

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

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

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

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

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

9. To add or edit vertical annotations, choose **Graph options: Add vertical annotation**:
 - a. To add a vertical annotation, choose **Add vertical annotation**.
 - b. For **Label**, enter a label for the annotation. To show only the date and time on the annotation, keep the **Label** field blank.
 - c. For **Date**, specify the date and time where the vertical annotation appears.
 - d. For **Fill**, specify whether to use fill shading before or after a vertical annotation, or between two vertical annotations. For example, choose **Before** or **After** for the corresponding area to be filled. If you specify **Between**, another **Date** field appears, and the area of the graph between the two values is filled.

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

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

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

10. To hide or change the position of the graph legend, hover over the title of the graph and choose **Widget actions, Edit**. Hover over **Legend** and choose **Hidden**, **Bottom**, or **Right**.
11. To customize the y-axis, choose **Graph options**. You can enter a custom label in **Label** under **Left Y Axis**. If the graph also displays values on the right y-axis, you can customize that label, too. You can also set minimums and maximums on the y-axis values, and the graph displays only the value range that you specify.
12. When you're finished with your changes, choose **Update widget**.

To temporarily hide metrics for a graph on a dashboard

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

Graph Metrics Manually on a CloudWatch Dashboard

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

To add a metric that you can't find in search to a graph

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

2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. The dashboard must already contain a graph where you want to add the metric. If it doesn't, create the graph and add any metric to it. For more information, see [Add or Remove a Graph from a CloudWatch Dashboard \(p. 20\)](#).
4. Choose **Actions, View/edit source**.

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

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

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

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

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

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

For more information about the formatting of metrics in JSON, see [Properties of a Metric Widget Object](#).

6. Choose **Update**.

Rename a Graph on a CloudWatch Dashboard

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

To rename a graph on a dashboard

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

Add or Remove a Text Widget from a CloudWatch Dashboard

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

To add a text widget to a dashboard

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

To edit a text widget on a dashboard

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

To remove a text widget from a dashboard

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

Add or Remove an Alarm from a CloudWatch Dashboard

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

To add an alarm to a dashboard

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, select the alarm to add, and then choose **Add to Dashboard**.
3. Select a dashboard, choose a widget type (**Line**, **Stacked area**, or **Number**) and then choose **Add to dashboard**.
4. To see your alarm on the dashboard, choose **Dashboards** in the navigation pane and select the dashboard.

5. (Optional) To temporarily make an alarm graph larger, select the graph.
6. (Optional) To change the widget type, hover over the title of the graph and choose **Widget actions** and then **Widget type**.

To remove an alarm from a dashboard

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

Use Live Data

You can choose whether your metric widgets display *live data*. Live data is data published within the last minute that has not been fully aggregated.

- If live data is turned **off**, only data points with an aggregation period of at least one minute in the past are shown. For example, when using 5-minute periods, the data point for 12:35 would be aggregated from 12:35 to 12:40, and displayed at 12:41.
- If live data is turned **on**, the most recent data point is shown as soon as any data is published in the corresponding aggregation interval. Each time you refresh the display, the most recent data point may change as new data within that aggregation period is published. If you use a cumulative statistic such as **Sum** or **Sample Count**, using live data may result in a dip at the end of your graph.

You can choose to enable live data for a whole dashboard, or for individual widgets on the dashboard.

To choose whether to use live data on your entire dashboard

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. To permanently turn on or off live data for all widgets on the dashboard, do the following:
 - a. Choose **Actions, Settings, Bulk update live data**.
 - b. Choose **Live Data on** or **Live Data off**, and choose **Set**.
4. To temporarily override the live data settings of each widget, do this: choose **Actions**, then under **Overrides**, next to **Live data**, do one of the following:
 - Choose **On** to temporarily turn on live data for all widgets.
 - Choose **Off** to temporarily turn off live data for all widgets.
 - Choose **Do not override** to preserve each widget's live data setting.

To choose whether to use live data on a single widget

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Select a widget, and choose **Actions, Edit**.
4. Choose the **Graph options** tab.
5. Select or clear the check box under **Live Data**.

Link and Unlink Graphs on a CloudWatch Dashboard

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

To link the graphs on a dashboard

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Choose **Actions** and then **Link graphs**.

To unlink the graphs on a dashboard

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Clear **Actions** and then **Link graphs**.

Add a Dashboard to Your Favorites List

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

To add a dashboard to the Favorites list

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards**.
3. Select the star symbol next to the dashboard to add it.

Change the Period Override Setting or Refresh Interval for the CloudWatch Dashboard

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

To change the period override options

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Actions**.
3. Under **Period override**, choose one of the following:
 - Choose **Auto** to have the period of the metrics on each graph automatically adapt to the dashboard's time range.
 - Choose **Do not override** to ensure that the period setting of each graph is always obeyed.
 - Choose one of the other options to cause graphs added to the dashboard to always adapt that chosen time as their period setting.

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

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

To change the dashboard refresh interval

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. On the **Refresh options** menu (upper-right corner), choose **10 Seconds**, **1 Minute**, **2 Minutes**, **5 Minutes**, or **15 Minutes**.

To automatically refresh the dashboard

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

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

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

Note

If you create a dashboard with graphs that contain close to 100 or more high-resolution metrics, we recommend that you set the time range to no longer than one hour to ensure good dashboard performance. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

To change the dashboard time range

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

Note

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

To change the dashboard time format

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

Using Amazon CloudWatch Metrics

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

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

Contents

- [Viewing Available Metrics \(p. 31\)](#)
- [Searching for Available Metrics \(p. 34\)](#)
- [Getting Statistics for a Metric \(p. 35\)](#)
- [Graphing Metrics \(p. 42\)](#)
- [Publishing Custom Metrics \(p. 50\)](#)
- [Using Metric Math \(p. 52\)](#)
- [Using Search Expressions in Graphs \(p. 63\)](#)

Viewing Available Metrics

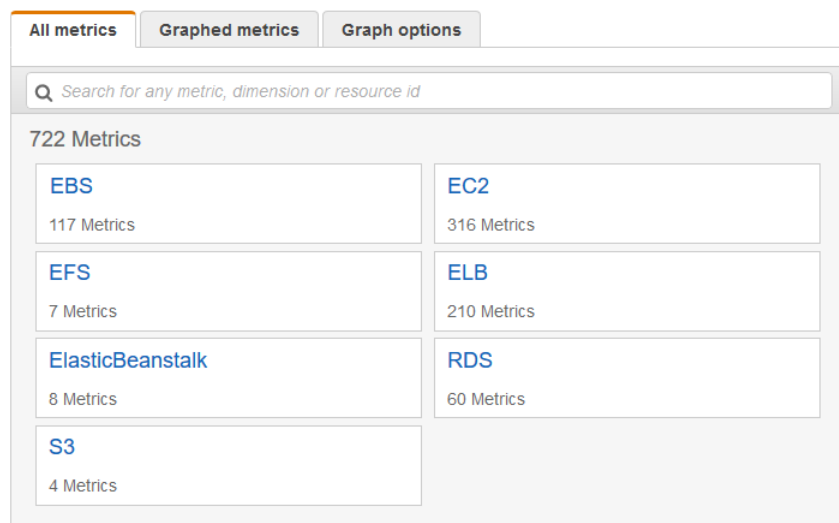
Metrics are grouped first by namespace, and then by the various dimension combinations within each namespace. For example, you can view all EC2 metrics, EC2 metrics grouped by instance, or EC2 metrics grouped by Auto Scaling group.

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

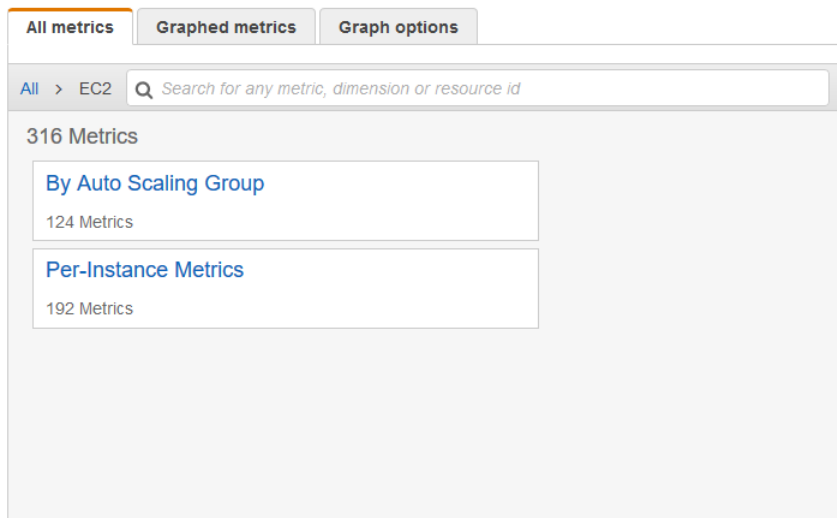
For a list of AWS services that send metrics to CloudWatch, see [AWS Services That Publish CloudWatch Metrics \(p. 378\)](#). From this page, you can also see the metrics and dimensions that are published by each of those services.

To view available metrics by namespace and dimension using the console

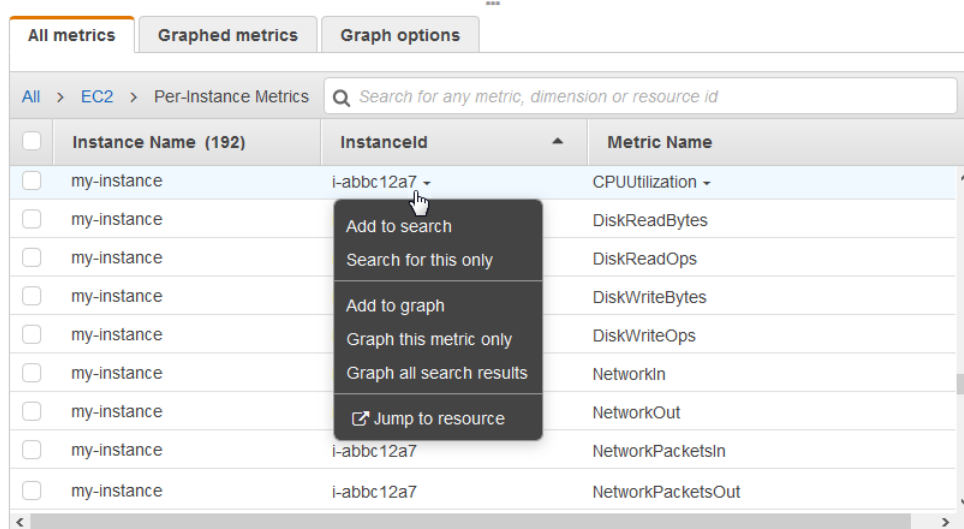
1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Select a metric namespace (for example, **EC2**).



4. Select a metric dimension (for example, **Per-Instance Metrics**).



5. The **All metrics** tab displays all metrics for that dimension in the namespace. You can do the following:
 - a. To sort the table, use the column heading.
 - b. To graph a metric, select the check box next to the metric. To select all metrics, select the check box in the heading row of the table.
 - c. To filter by resource, choose the resource ID and then choose **Add to search**.
 - d. To filter by metric, choose the metric name and then choose **Add to search**.



To view available metrics by namespace, dimension, or metric using the AWS CLI

Use the `list-metrics` command to list CloudWatch metrics. For a list of the namespaces, metrics, and dimensions for all services that publish metrics, see [AWS Services That Publish CloudWatch Metrics \(p. 378\)](#).

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

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

The following is example output.

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

```
{
  {
    "Name": "InstanceId",
    "Value": "i-1234567890abcdef0"
  }
},
"MetricName": "NetworkIn"
},
...
]
```

To list all the available metrics for a specified resource

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

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

To list a metric for all resources

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

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

Searching for Available Metrics

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

To search for available metrics in CloudWatch

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. In the search field on the **All metrics** tab, enter a search term, such as a metric name, namespace, dimension name or value, or resource name. This shows you all of the namespaces with metrics with this search term.

For example, if you search for **volume**, this shows the namespaces that contain metrics with this term in their name.

For more information on search, see [Using Search Expressions in Graphs](#) (p. 63)

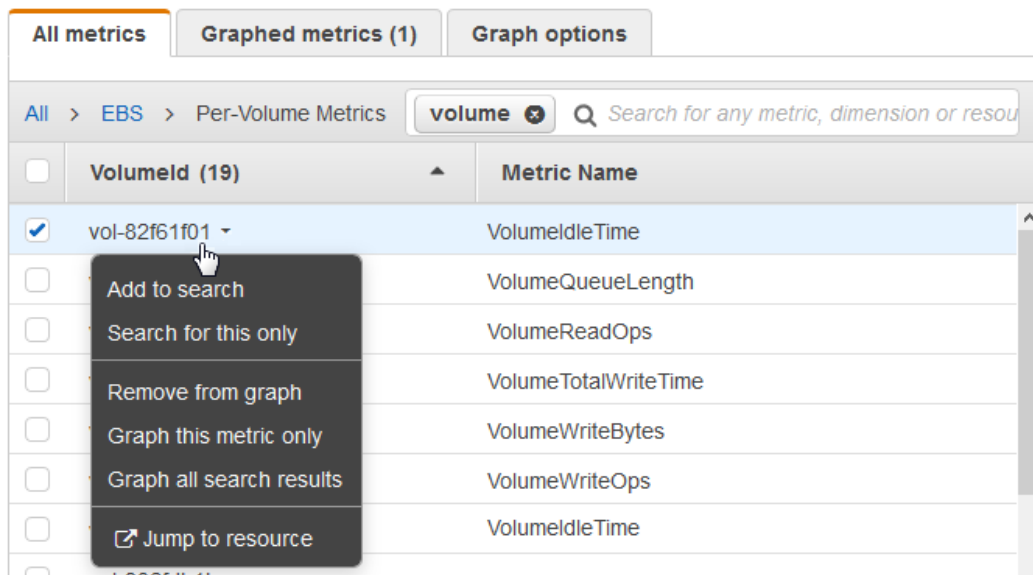
4. To graph all the search results, choose **Graph search**

or

Select a namespace to view the metrics from that namespace. You can then do the following:

- a. To graph one or more metrics, select the check box next to each metric. To select all metrics, select the check box in the heading row of the table.
- b. To refine your search, hover over a metric name and choose **Add to search** or **Search for this only**.
- c. To view one of the resources on its console, choose the resource ID and then choose **Jump to resource**.
- d. To view help for a metric, select the metric name and choose **What is this?**

The selected metrics appear on the graph.



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

Getting Statistics for a Metric

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

Examples

- [Getting Statistics for a Specific Resource \(p. 35\)](#)
- [Aggregating Statistics Across Resources \(p. 38\)](#)
- [Aggregating Statistics by Auto Scaling Group \(p. 40\)](#)
- [Aggregating Statistics by Amazon Machine Image \(AMI\) \(p. 41\)](#)

Getting Statistics for a Specific Resource

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

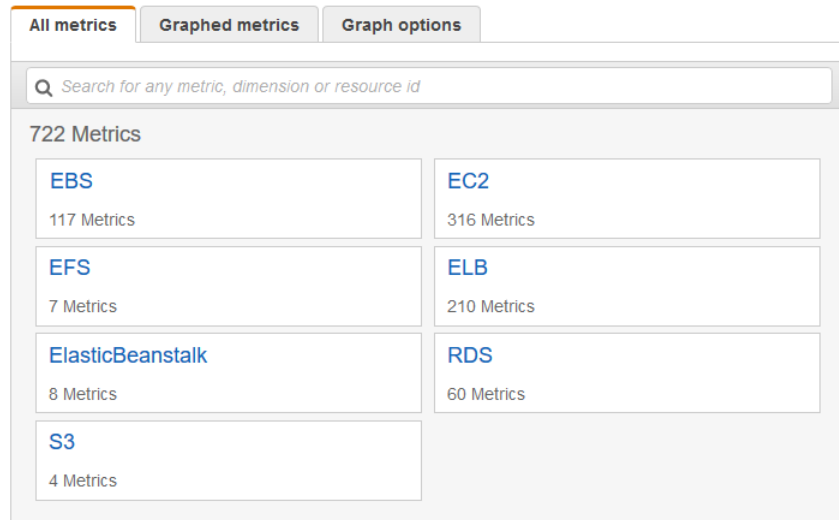
Requirements

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

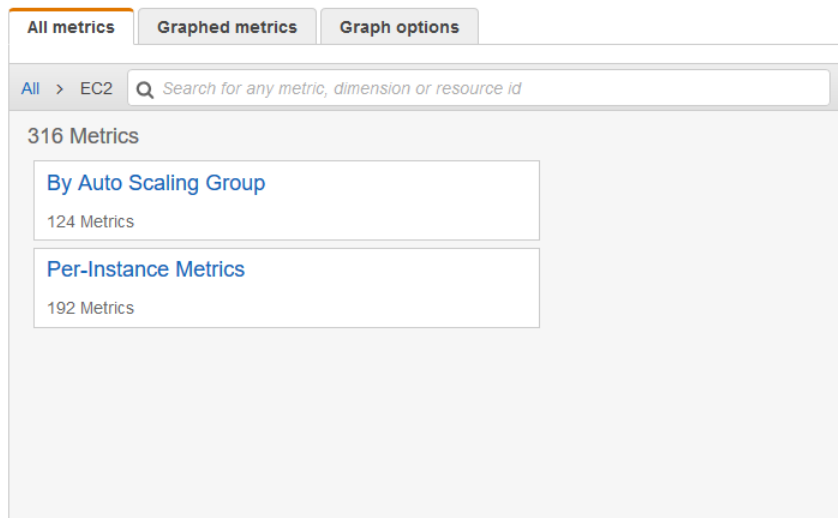
To display the average CPU utilization for a specific instance using the console

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

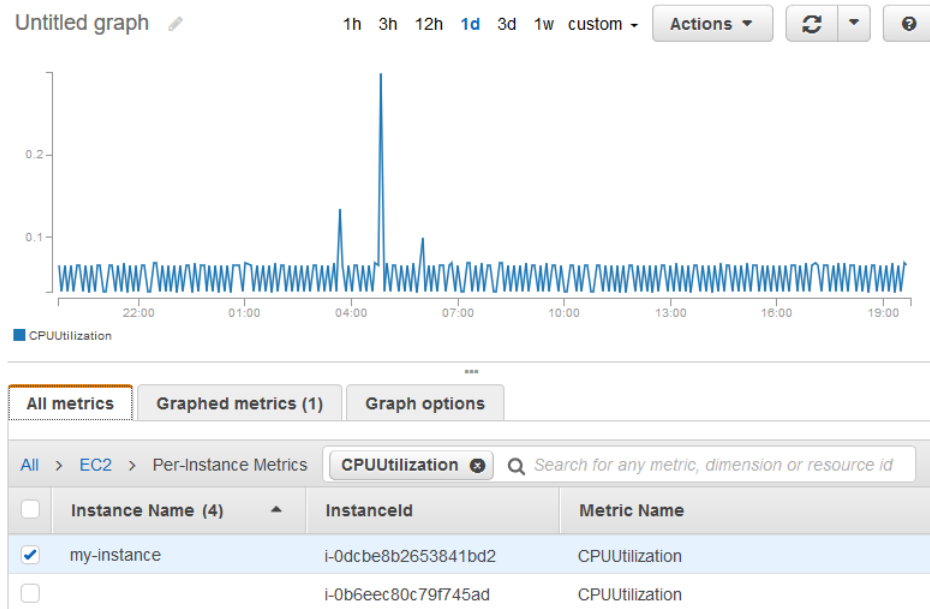
2. In the navigation pane, choose **Metrics**.
3. Select the **EC2** metric namespace.



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



5. In the search field, enter **CPUtilization** and press Enter. Select the row for the specific instance, which displays a graph for the **CPUtilization** metric for the instance. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.



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

All metrics Graphed metrics (1) Graph options

Label	Namespace	Dimensions	Metric Name	Statistic	Period
CPUUtilization	AWS/EC2	Dimensions (1)	CPUUtilization	Average	5 Minutes

Average
 Minimum
 Maximum
 Sum
 Data Samples
 p99
 p95
 p90
 p50
 p10
 Custom percentile...

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

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

Use the `get-metric-statistics` command as follows to get the CPUUtilization metric for the specified instance.

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-1234567890abcdef0 --statistics Maximum \
```

```
--start-time 2016-10-18T23:18:00 --end-time 2016-10-19T23:18:00 --period 360
```

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

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

Aggregating Statistics Across Resources

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

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

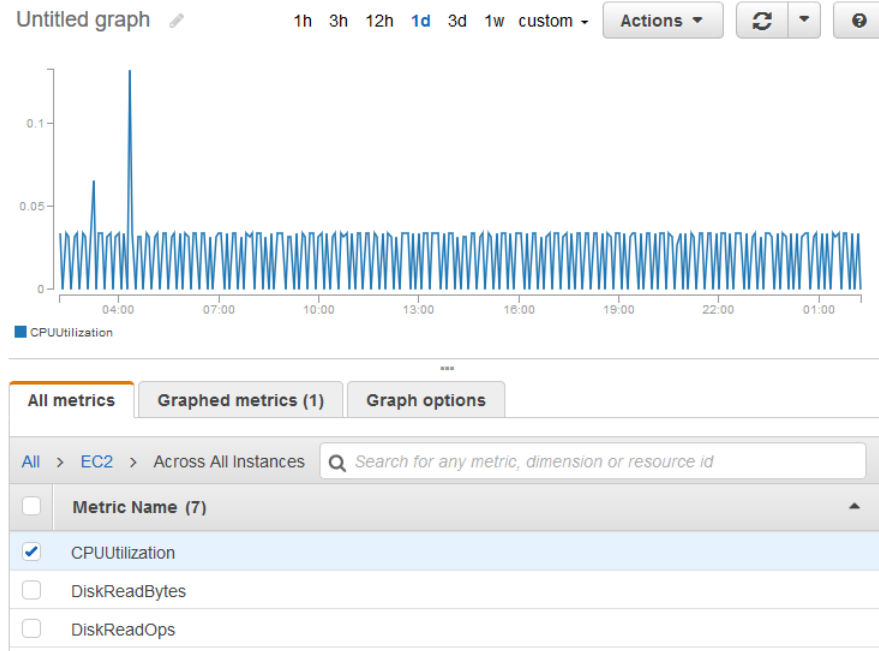
This example shows you how to get the average CPU usage for your EC2 instances. Because no dimension is specified, CloudWatch returns statistics for all dimensions in the AWS/EC2 namespace. To get statistics for other metrics, see [AWS Services That Publish CloudWatch Metrics](#) (p. 378).

Important

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

To display average CPU utilization for your EC2 instances

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose the **EC2** namespace and choose **Across All Instances**.
4. Select the row that contains `CPUUtilization`, which displays a graph for the metric for all your EC2 instances. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.



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

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

Use the [get-metric-statistics](#) command as follows:

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

The following is example output:

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

```
    ...  
  ],  
  "Label": "CPUUtilization"  
}
```

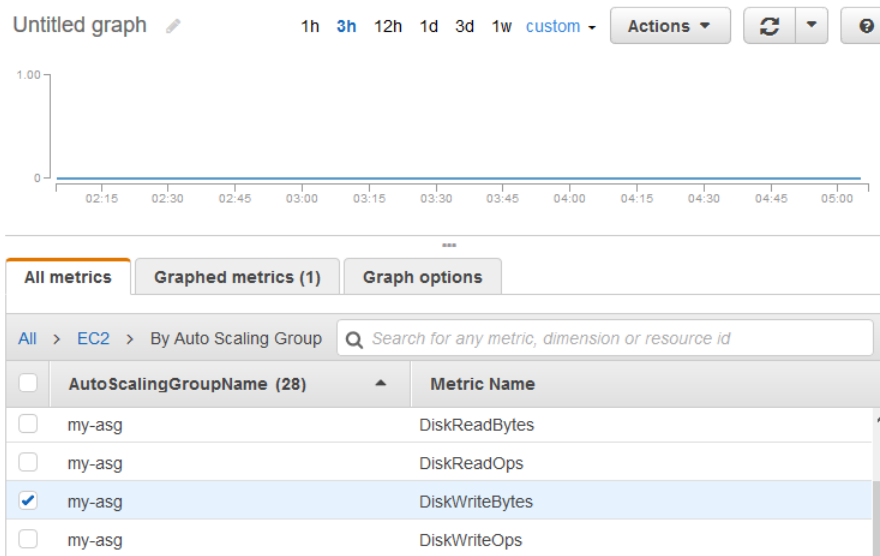
Aggregating Statistics by Auto Scaling Group

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

This example shows you how to get the total bytes written to disk for one Auto Scaling group. The total is computed for 1-minute periods for a 24-hour interval across all EC2 instances in the specified Auto Scaling group.

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

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose the **EC2** namespace and then choose **By Auto Scaling Group**.
4. Select the row for the **DiskWriteBytes** metric and the specific Auto Scaling group, which displays a graph for the metric for the instances in the Auto Scaling group. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.



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

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

Use the [get-metric-statistics](#) command as follows.

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


The following is example output.

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

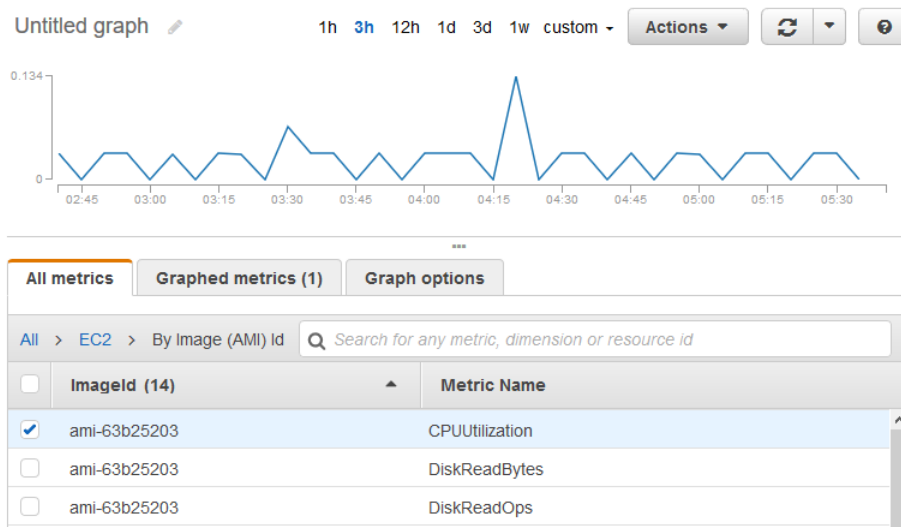
Aggregating Statistics by Amazon Machine Image (AMI)

You can aggregate statistics for the EC2 instances that have detailed monitoring enabled. Instances that use basic monitoring aren't included. For more information, see [Enable or Disable Detailed Monitoring for Your Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

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

To display the average CPU utilization by AMI using the console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose the **EC2** namespace and then choose **By Image (AMI) Id**.
4. Select the row for the **CPUUtilization** metric and the specific AMI, which displays a graph for the metric for the specified AMI. To change the name of the graph, choose the pencil icon. To change the time range, select one of the predefined values or choose **custom**.



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

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

Use the [get-metric-statistics](#) command as follows.

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

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

```
{
  "Datapoints": [
    {
      "Timestamp": "2016-10-10T07:00:00Z",
      "Average": 0.041000000000000009,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2016-10-10T14:00:00Z",
      "Average": 0.079579831932773085,
      "Unit": "Percent"
    },
    {
      "Timestamp": "2016-10-10T06:00:00Z",
      "Average": 0.0360000000000000011,
      "Unit": "Percent"
    },
    ...
  ],
  "Label": "CPUUtilization"
}
```

Graphing Metrics

Use the CloudWatch console to graph metric data generated by other AWS services. This makes it more efficient to see the metric activity on your services. The following procedures describe how to graph metrics in CloudWatch.

Contents

- [Graphing a Metric \(p. 43\)](#)
- [Using Dynamic Labels \(p. 46\)](#)
- [Modifying the Time Range or Time Zone Format for a Graph \(p. 47\)](#)
- [Modifying the Y-Axis for a Graph \(p. 48\)](#)
- [Creating an Alarm from a Metric on a Graph \(p. 49\)](#)

Graphing a Metric

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

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

You can view your data at different levels of detail. For example, you can choose a one-minute view, which can be useful when troubleshooting. Or, choose a less detailed, one-hour view. That can be useful when viewing a broader time range (for example, 3 days) so that you can see trends over time. For more information, see [Periods \(p. 6\)](#).

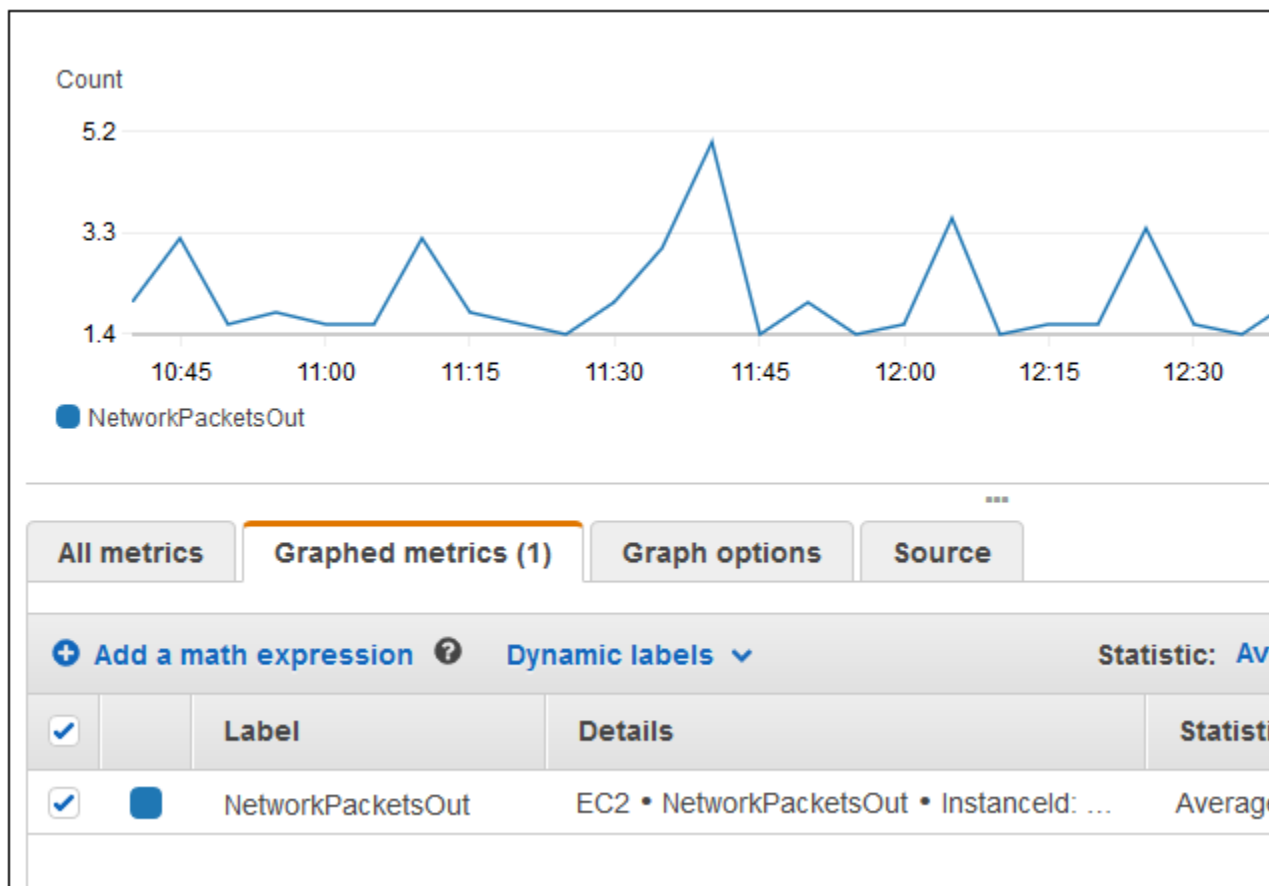
Creating a Graph

To graph a metric

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

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

4. Select one of the results for your search to view the metrics.
5. To graph one or more metrics, select the check box next to each metric. To select all metrics, select the check box in the heading row of the table.
6. (Optional) To add an anomaly detection band that shows expected values for the metric, choose the anomaly detection icon under **Actions** next to the metric.



CloudWatch uses up to two weeks of the metric's recent historical data to calculate a model for expected values. It then displays this range of expected values as a band on the graph. CloudWatch adds a new row under the metric to display the anomaly detection band math expression, labeled **ANOMALY_DETECTION_BAND**. If recent historical data exists, you immediately see a preview anomaly detection band, which is an approximation of the anomaly detection band generated by the model. It takes up to 15 minutes for the actual anomaly detection band to appear.

By default, CloudWatch creates the upper and lower bounds of the band of expected values with a default value of 2 for the band threshold. To change this number, change the value at the end of the formula under **Details** for the band.

- (Optional) Choose **Edit model** to change how the anomaly detection model is calculated. You can exclude past and future time periods from being used in the training for calculating the model. It is critical to exclude unusual events system as system outage, deployments, and holidays from the training data. You can also specify the time zone to use for the model for daylight saving time changes.

For more information, see [Using CloudWatch Anomaly Detection \(p. 141\)](#).

To hide the model from the graph, remove the checkmark from the line with the **ANOMALY_DETECTION_BAND** function or choose the x icon. To delete the model entirely, choose **Edit model, Delete model**.

7. (Optional) As you choose metrics to graph, specify a dynamic label to appear on the graph legend for each metric. Dynamic labels display a statistic about the metric, and automatically update when

the dashboard or graph is refreshed. To add a dynamic label, choose **Graphed metrics, Dynamic labels**.

By default, the dynamic values that you add to the label appear at the beginning of the label. You can then choose the **Label** value for the metric to edit the label. For more information, see [Using Dynamic Labels \(p. 46\)](#).

8. To view more information about the metric being graphed, pause the mouse over the legend.
9. Horizontal annotations can help graph users more efficiently see when a metric has spiked to a certain level, or whether the metric is within a predefined range. To add a horizontal annotation, choose **Graph options** and then **Add horizontal annotation**:
 - a. For **Label**, enter a label for the annotation.
 - b. For **Value**, enter the metric value where the horizontal annotation appears.
 - c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose **Above** or **Below** for the corresponding area to be filled. If you specify **Between**, another **Value** field appears, and the area of the graph between the two values is filled.
 - d. For **Axis**, specify whether the numbers in **Value** refer to the metric associated with the left Y-axis or the right Y-axis, if the graph includes multiple metrics.

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

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

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

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

10. To get a URL for your graph, choose **Actions, Share**. Copy the URL to save or share.
11. To add your graph to a dashboard, choose **Actions, Add to dashboard**.

Updating a Graph

To update your graph

1. To change the name of the graph, choose the pencil icon.
2. To change the time range, select one of the predefined values or choose **custom**. For more information, see [Modifying the Time Range or Time Zone Format for a Graph \(p. 47\)](#).
3. To change the statistic, choose the **Graphed metrics** tab. Choose the column heading or an individual value and then choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, **p95.45**).
4. To change the period, choose the **Graphed metrics** tab. Choose the column heading or an individual value and then choose a different value.
5. To add a horizontal annotation, choose **Graph options** and then **Add horizontal annotation**:
 - a. For **Label**, enter a label for the annotation.
 - b. For **Value**, enter the metric value where the horizontal annotation appears.
 - c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose **Above** or **Below** for the corresponding area to be filled. If you specify **Between**, another **Value** field appears, and the area of the graph between the two values is filled.
 - d. For **Axis**, specify whether the numbers in **Value** refer to the metric associated with the left y-axis or the right y-axis, if the graph includes multiple metrics.

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

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

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

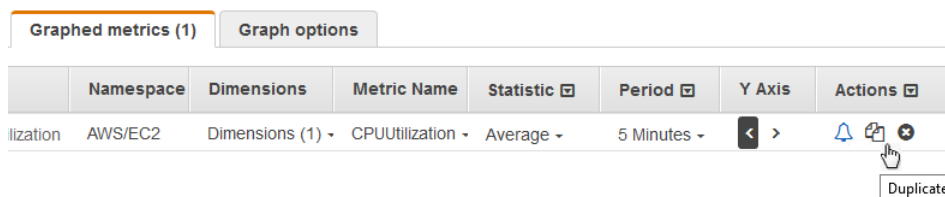
To delete an annotation, choose **x** in the **Actions** column.

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

Duplicating a Metric

To duplicate a metric

1. Choose the **Graphed metrics** tab.
2. For **Actions**, choose the **Duplicate** icon.



3. Update the duplicate metric as needed.

Using Dynamic Labels

You can use dynamic labels with your graphs. Dynamic labels add a dynamically updated value to the label for the selected metric. The values displayed can include the average, maximum, minimum, sum, or the most recent data point. The dynamic value shown in the label is derived from the time range currently shown on the graph. This part of the label automatically updates when either the dashboard or the graph is refreshed.

If you use a dynamic label with a search expression, the dynamic label applies to every metric returned by the search.

The CloudWatch console makes it easy for you to add a dynamic value to a label. You can then further edit the label, to change the position of the dynamic value within the label or make other customizations.

Dynamic Label Syntax

Within a dynamic label, the dynamic values can include the following:

Variable	Description
\${AVG}	The average of the values in the time range currently shown in the graph.
\${LAST}	The most recent of the values in the time range currently shown in the graph.

Variable	Description
\${LABEL}	Represents the default label for a metric
\${MAX}	The maximum of the values in the time range currently shown in the graph.
\${MIN}	The minimum of the values in the time range currently shown in the graph.
\${SUM}	The sum of the values in the time range currently shown in the graph.

For example, suppose you have a search expression `SEARCH(' {AWS/Lambda, FunctionName} Errors ', 'Sum', 300)`, which finds the `Errors` for each of your Lambda functions. If you set the label to be `[max: ${MAX} Errors for Function Name ${LABEL}]`, the label for each metric is `[max: number Errors for Function Name Name]`.

You can add up to five dynamic statistical values to a label. You can use the `${LABEL}` placeholder only once within each label.

Modifying the Time Range or Time Zone Format for a Graph

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

Setting a Relative Time Range

You can set a relative time range for your graph.

To specify a relative time range for a graph

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

Setting an Absolute Time Range

You can set an absolute time range for your graph.

To specify an absolute time range for a graph

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose the **custom** menu and then choose **Absolute**. Use the calendar picker or the text fields to specify the time range.

Setting the Time Zone Format

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

To specify the time zone for a graph

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose the **custom** menu and then choose **UTC** or **Local timezone**.

Zooming In on a Graph

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

To zoom in on a graph

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose and drag on the graph area and then release the drag.
4. To reset a zoomed-in graph, choose the **Reset zoom** icon.

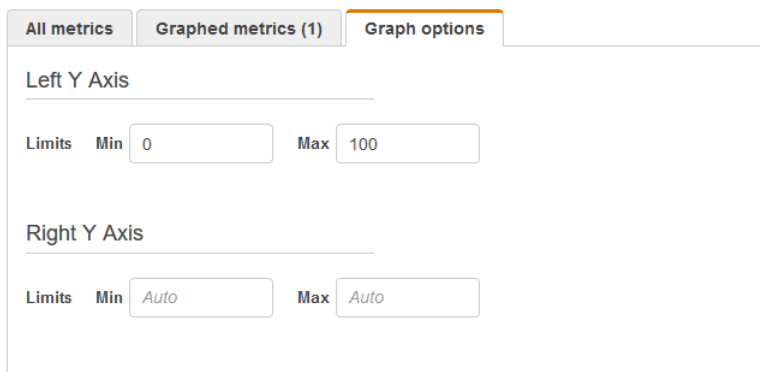
Modifying the Y-Axis for a Graph

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

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

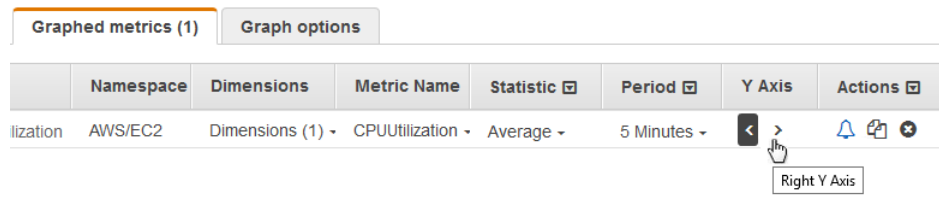
To modify the y-axis on a graph

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Select a metric namespace (for example, **EC2**) and then a metric dimension (for example, **Per-Instance Metrics**).
4. The **All metrics** tab displays all metrics for that dimension in that namespace. To graph a metric, select the check box next to the metric.
5. On the **Graph options** tab, specify the **Min** and **Max** values for **Left Y Axis**. The value of **Min** can't be greater than the value of **Max**.



The screenshot shows the 'Graph options' tab in the CloudWatch console. It contains two sections: 'Left Y Axis' and 'Right Y Axis'. Each section has a 'Limits' label and two input fields: 'Min' and 'Max'. In the 'Left Y Axis' section, the 'Min' field is set to '0' and the 'Max' field is set to '100'. In the 'Right Y Axis' section, both the 'Min' and 'Max' fields are set to 'Auto'. The 'Graphed metrics (1)' tab is also visible, indicating a metric is selected for graphing.

6. To create a second y-axis, specify the **Min** and **Max** values for **Right Y Axis**.
7. To switch between the two y-axes, choose the **Graphed metrics** tab. For **Y Axis**, choose **Left Y Axis** or **Right Y Axis**.

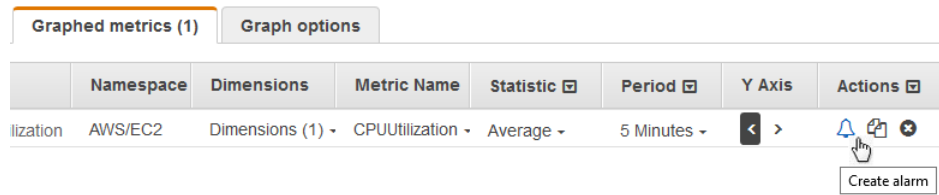


Creating an Alarm from a Metric on a Graph

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

To create an alarm from a metric on a graph

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Select a metric namespace (for example, **EC2**) and then a metric dimension (for example, **Per-Instance Metrics**).
4. The **All metrics** tab displays all metrics for that dimension in that namespace. To graph a metric, select the check box next to the metric.
5. To create an alarm for the metric, choose the **Graphed metrics** tab. For **Actions**, choose the alarm icon.



6. Under **Conditions**, choose **Static** or **Anomaly detection** to specify whether to use a static threshold or anomaly detection model for the alarm.

Depending on your choice, enter the rest of the data for the alarm conditions.

7. Choose **Additional configuration**. For **Datapoints to alarm**, specify how many evaluation periods (data points) must be in the **ALARM** state to trigger the alarm. If the two values here match, you create an alarm that goes to **ALARM** state if that many consecutive periods are breaching.

To create an M out of N alarm, specify a lower number for the first value than you specify for the second value. For more information, see [Evaluating an Alarm \(p. 72\)](#).

8. For **Missing data treatment**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
9. Choose **Next**.
10. Under **Notification**, select an SNS topic to notify when the alarm is in **ALARM** state, **OK** state, or **INSUFFICIENT_DATA** state.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

To have the alarm not send notifications, choose **Remove**.

11. To have the alarm perform Auto Scaling or EC2 actions, choose the appropriate button and choose the alarm state and action to perform.

12. When finished, choose **Next**.
13. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.
14. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

Publishing Custom Metrics

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

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

Topics

- [High-Resolution Metrics \(p. 50\)](#)
- [Using Dimensions \(p. 50\)](#)
- [Publishing Single Data Points \(p. 51\)](#)
- [Publishing Statistic Sets \(p. 52\)](#)
- [Publishing the Value Zero \(p. 52\)](#)

High-Resolution Metrics

Each metric is one of the following:

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

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

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

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

Using Dimensions

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

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

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

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

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

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

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

To see what dimensions are defined for a metric, use the [list-metrics](#) command.

Publishing Single Data Points

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

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

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

Note

When you create a metric, it can take up to 2 minutes before you can retrieve statistics for the new metric using the [get-metric-statistics](#) command. However, it can take up to 15 minutes before the new metric appears in the list of metrics retrieved using the [list-metrics](#) command.

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

You can use the [get-metric-statistics](#) command to retrieve statistics based on the data points that you published.

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

The following is example output.

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

Publishing Statistic Sets

You can aggregate your data before you publish to CloudWatch. When you have multiple data points per minute, aggregating data minimizes the number of calls to **put-metric-data**. For example, instead of calling **put-metric-data** multiple times for three data points that are within 3 seconds of each other, you can aggregate the data into a statistic set that you publish with one call, using the `--statistic-values` parameter.

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

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

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

Publishing the Value Zero

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

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

Using Metric Math

Metric math enables you to query multiple CloudWatch metrics and use math expressions to create new time series based on these metrics. You can visualize the resulting time series on the CloudWatch console and add them to dashboards. Using AWS Lambda metrics as an example, you could divide the `Errors` metric by the `Invocations` metric to get an error rate. Then add the resulting time series to a graph on your CloudWatch dashboard.

You can also perform metric math programmatically, using the `GetMetricData` API operation. For more information, see [GetMetricData](#).

Adding a Math Expression to a CloudWatch Graph

You can add a math expression to a graph on your CloudWatch dashboard. Each graph is limited to using a maximum of 500 metrics and expressions, so you can add a math expression only if the graph has 499 or fewer metrics. This applies even if not all the metrics are displayed on the graph.

To add a math expression to a graph

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Create or edit a graph. There needs to be at least one metric in the graph.
3. Choose **Graphed metrics**.
4. Choose **Math expression, Start with empty expression**. A new line appears for the expression.
5. In the new line, under the **Details** column, enter the math expression. The tables in the Metric Math Syntax and Functions section list the functions that you can use in the expression.

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

You can change the value of **Id**. It can include numbers, letters, an underscore, and must start with a lowercase letter. Changing the value of **Id** to a more meaningful name can also make a graph easier to understand; for example, changing **m1** and **m2** to **errors** and **requests**.

Tip

Choose the down arrow next to **Math Expression** to see a list of supported functions, which you can use when creating your expression.

6. For the **Label** column of the expression, enter a name that describes what the expression is calculating.

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

When one expression produces an array of time series on the graph, you can't change the colors used for each of those time series.

7. After you have added the desired expressions, you can simplify the graph by hiding some of the original metrics. To hide a metric or expression, clear the check box to the left of the **Id** field.

Metric Math Syntax and Functions

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

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

Data Type Abbreviations

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

- **S** represents a scalar number, such as 2, -5, or 50.25
- **TS** is a time series (a series of values for a single CloudWatch metric over time): for example, the `CPUUtilization` metric for instance `i-1234567890abcdef0` over the last 3 days
- **TS[]** is an array of time series, such as the time series for multiple metrics

The METRICS() Function

The **METRICS()** function returns all the metrics in the request. Math expressions aren't included.

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

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

Basic Arithmetic Functions

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

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

Comparison and Logical Operators

You can use comparison and logical operators with either a pair of time series or a pair of single scalar values. When you use a comparison operator with a pair of time series, the operators return a time series

where each data point is either 0 (false) or 1 (true). If you use a comparison operator on a pair of scalar values, a single scalar value is returned, either 0 or 1.

When comparison operators are used between two time series, and only one of the time series has a value for a particular time stamp, the function treats the missing value in the other time series as **0**.

You can use logical operators in conjunction with comparison operators, to create more complex functions.

The following table lists the operators that are supported.

Type of Operator	Supported Operators
Comparison operators	== != <= >= < >
Logical operators	AND or && OR or

To see how these operators are used, suppose we have two time series: **metric1** has values of [30, 20, 0, 0] and **metric2** has values of [20, -, 20, -] where - indicates that there is no value for that timestamp.

Expression	Output
(metric1 < metric2)	0, 0, 1, 0
(metric1 >= 30)	1, 0, 0, 0
(metric1 > 15 AND metric2 > 15)	1, 0, 0, 0

Functions Supported for Metric Math

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

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

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

Function	Argument	Return Type*	Description	Examples
ABS	TS TS[]	TS TS[]	Returns the absolute value of each data point.	ABS(m1-m2) MIN(ABS([m1, m2])) ABS(METRICS())
ANOMALY_DETECTION_BAND	TS, S	TS[]	Returns an anomaly detection band for the specified metric. The band consists of two time series, one representing the upper limit of the "normal" expected value of the metric, and the other representing the lower limit. The function can take two arguments. The first is the ID of the metric to create the band for. The second argument is the number of standard deviations to use for the band. If you don't specify this argument, the default of 2 is used. For more information, see Using CloudWatch Anomaly Detection (p. 141) .	ANOMALY_DETECTION_BAND(m1) ANOMALY_DETECTION_BAND(m1,4)
AVG	TS TS[]	S TS	The AVG of a single time series returns a scalar representing the average of all the data points in the metric. The AVG of an array of time series returns a single time series. Missing values are treated as 0.	SUM([m1,m2])/AVG(m2) AVG(METRICS())
CEIL	TS TS[]	TS TS[]	Returns the ceiling of each metric. The ceiling is the smallest integer greater than or equal to each value.	CEIL(m1) CEIL(METRICS()) SUM(CEIL(METRICS()))
FILL	TS, TS/ S TS[], TS/S	TS TS[]	Fills the missing values of a metric with the specified filler value when the metric values are sparse.	FILL(m1,10) FILL(METRICS(), 0) FILL(m1, MIN(m1))
FIRST LAST	TS[]	TS	Returns the first or last time series from an array of time series. This is useful when used with the SORT function. It can also be used to get the high and	IF(FIRST(SORT(METRICS(), AVG, DESC))>100, 1, 0) Looks at the top metric from an array, which is sorted by AVG. It then returns a 1 or 0 for each datapoint,

Function	Argument	Return Type*	Description	Examples
			low thresholds from the ANOMALY_DETECTION_BAND function.	depending on whether that datapoint value is more than 100. LAST(ANOMALY_DETECTION_BAND(m1)) returns the lower bound of the anomaly prediction band.
FLOOR	TS TS[]	TS TS[]	Returns the floor of each metric. The floor is the largest integer less than or equal to each value.	FLOOR(m1) FLOOR(METRICS())
IF	IF expression	TS	Use IF along with a comparison operator to filter out data points from a time series, or create a mixed time-series composed of multiple collated time series. For more information, see Using IF Expressions (p. 61) .	
INSIGHT_RULE_METRIC	INSIGHT_RULE_METRIC (metricName)	INSIGHT_RULE_METRIC	Use INSIGHT_RULE_METRIC to extract statistics from a rule in Contributor Insights. For more information, see Graphing Metrics Generated by Rules (p. 152) .	
MAX	TS TS[]	S TS	The MAX of a single time series returns a scalar representing the maximum value of all data points in the metric. The MAX value of an array of time series returns a single time series.	MAX(m1)/m1 MAX(METRICS())
METRIC_COUNT	TS[]	S	Returns the number of metrics in the time series array.	m1/ METRIC_COUNT(METRICS())

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

Function	Argument	Return Type*	Description	Examples
REMOVE_EMPTY	TS[]	TS[]	Removes any time series that have no datapoints from an array of time series. The result is an array of time series where each time series contains at least one datapoint.	REMOVE_EMPTY(METRICS())
SEARCH	Search expression	One or more TS	Returns one or more time series that match a search criteria that you specify. The SEARCH function enables you to add multiple related time series to a graph with one expression. The graph is dynamically updated to include new metrics that are added later and match the search criteria. For more information, see Using Search Expressions in Graphs (p. 63) .	
SERVICE_QUOTA	TS that is a usage metric	TS	Returns the service quota for the given usage metric. You can use this to visualize how your current usage compares to the quota, and to set alarms that warn you when you approach the quota. For more information, see Service Quotas Integration and Usage Metrics (p. 409) .	

Function	Argument	Return Type*	Description	Examples
SLICE	(TS[], S, S) or (TS[], S)	TS[] TS	<p>Retrieves part of an array of time series. This is especially useful when combined with SORT. For example, you can exclude the top result from an array of time series.</p> <p>You can use two scalar arguments to define the set of time series that you want returned. The two scalars define the start (inclusive) and end (exclusive) of the array to return. The array is zero-indexed, so the first time series in the array is time series 0. Alternatively, you can specify just one value, and CloudWatch returns all time series starting with that value.</p>	<p>SLICE(SORT(METRICS()), SUM, DESC, 0, 10) returns the 10 metrics from the array of metrics in the request that have the highest SUM value.</p> <p>SLICE(SORT(METRICS()), AVG, ASC, 5) sorts the array of metrics by the AVG statistic, then returns all the time series except for the 5 with the lowest AVG.</p>
SORT	(TS[], FUNCTION, SORT_ORDER) (TS[], FUNCTION, SORT_ORDER, S)	TS[]	<p>Sorts an array of time series according to the function you specify. The function you use can be AVG, MIN, MAX, or SUM. The sort order can be either ASC for ascending (lowest values first) or DESC to sort the higher values first. You can optionally specify a number after the sort order which acts as a limit. For example, specifying a limit of 5 returns only the top 5 time series from the sort.</p> <p>When this math function is displayed on a graph, the labels for each metric in the graph are also sorted and numbered.</p>	<p>SORT(METRICS(), AVG, DESC, 10) calculates the average value of each time series, sorts the time series with the highest values at the beginning of the sort, and returns only the 10 time series with the highest averages.</p> <p>SORT(METRICS(), MAX, ASC) sorts the array of metrics by the MAX statistic, then returns all of them in ascending order.</p>
STDDEV	TS TS[]	S TS	The STDDEV of a single time series returns a scalar representing the standard deviation of all data points in the metric. The STDDEV of an array of time series returns a single time series.	<p>m1/STDDEV(m1)</p> <p>STDDEV(METRICS())</p>

Function	Argument	Return Type*	Description	Examples
SUM	TS TS[]	S TS	The SUM of a single time series returns a scalar representing the sum of the values of all data points in the metric. The SUM of an array of time series returns a single time series.	SUM(METRICS())/SUM(m1) SUM([m1,m2]) SUM(METRICS("errors"))/SUM(METRICS("requests"))*100

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

Using IF Expressions

Use **IF** along with a comparison operator to filter out data points from a time series, or create a mixed time-series composed of multiple collated time series.

IF uses the following arguments:

```
IF(condition, trueValue, falseValue)
```

The condition evaluates to FALSE if the value of the condition data point is 0, and to TRUE if the value of the condition is any other value, whether that value is positive or negative. If the condition is a time series, it is evaluated separately for every timestamp.

The following lists the valid syntaxes. For each of these syntaxes, the output is a single time series.

- **IF(TS *Comparison Operator* S, S | TS, S / TS)**
- **IF(TS, TS, TS)**
- **IF(TS, S, TS)**
- **IF(TS, TS, S)**
- **IF(TS, S, S)**
- **IF(S, TS, TS)**

The following sections provide more details and examples for these syntaxes.

IF(TS *Comparison Operator* S, scalar2 | metric2, scalar3 | metric3)

The corresponding output time series value:

- has the value of **scalar2** or **metric2**, if **TS *Comparison Operator* S** is TRUE
- has the value of **scalar3** or **metric3**, if **TS *Comparison Operator* S** is FALSE
- is an empty time series, if the corresponding data point of does not exist in **metric3**, or if **scalar3/metric3** is omitted from the expression

IF(metric1, metric2, metric3)

For each data point of **metric1**, the corresponding output time series value:

- has the value of **metric2**, if the corresponding data point of **metric1** is TRUE.

- has the value of **metric3**, if the corresponding data point of **metric1** is FALSE.
- has the value of **0**, if the corresponding data point of **metric1** is TRUE and the corresponding data point does not exist in **metric2**.
- is dropped, if the corresponding data point of **metric1** is FALSE and the corresponding data point does not exist in **metric3** or if **metric3** is omitted from the expression.

The following table shows an example for this syntax.

Metric or Function	Values
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
(metric3)	[0, 0, 20, -, 20]
IF(metric1, metric2, metric3)	[30, 0, 20, -, -]

IF(metric1, scalar2, metric3)

For each data point of **metric1**, the corresponding output time series value:

- has the value of **scalar2**, if the corresponding data point of **metric1** is TRUE.
- has the value of **metric3**, if the corresponding data point of **metric1** is FALSE.
- is dropped, if the corresponding data point of **metric1** is FALSE and the corresponding data point does not exist on **metric3**, or if **metric3** is omitted from the expression.

Metric or Function	Values
(metric1)	[1, 1, 0, 0, -]
scalar2	5
(metric3)	[0, 0, 20, -, 20]
IF(metric1, scalar2, metric3)	[5, 5, 20, -, -]

IF(metric1, metric2, scalar3)

For each data point of **metric1**, the corresponding output time series value:

- has the value of **metric2**, if the corresponding data point of **metric1** is TRUE.
- has the value of **scalar3**, if the corresponding data point of **metric1** is FALSE.
- has the value of **0**, if the corresponding data point of **metric1** is TRUE and the corresponding data point does not exist in **metric2**.

Metric or Function	Values
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]

Metric or Function	Values
scalar3	5
IF(metric1, metric2, scalar3)	[30, 0, 5, 5, 5]

IF(scalar1, metric2, metric3)

The corresponding output time series value:

- has the value of **metric2**, if **scalar1** is TRUE.
- has the value of **metric3**, if **scalar1** is FALSE.
- is an empty time series, if **metric3** is omitted from the expression.

Use Case Examples for IF Expressions

The following examples illustrate the possible uses of the **IF** function.

- To display only the low values of a metric:

IF(metric1<400, metric1)

- To change each data point in a metric to one of two values, to show relative highs and lows of the original metric:

IF(metric1<400, 10, 2)

- To display a 1 for each timestamp where latency is over the threshold, and display a 0 for all other data points:

IF(latency>threshold, 1, 0)

Using Metric Math with the GetMetricData API Operation

You can use `GetMetricData` to perform calculations using math expressions, and also retrieve large batches of metric data in one API call. For more information, see [GetMetricData](#).

Using Search Expressions in Graphs

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

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

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

Topics

- [CloudWatch Search Expression Syntax \(p. 64\)](#)

- [CloudWatch Search Expression Examples \(p. 68\)](#)
- [Creating a CloudWatch Graph with a Search Expression \(p. 69\)](#)

CloudWatch Search Expression Syntax

A valid search expression has the following format.

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

For example:

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

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

You don't need to use quote marks inside the metric schema unless a namespace or dimension name includes spaces or non-alphanumeric characters. In that case, you must enclose the name that contains those characters with double quotes.

- The `SearchTerm` is also optional, but a valid search must contain either the metric schema, the `SearchTerm`, or both. The `SearchTerm` usually contains one or more metric names or dimension values. The `SearchTerm` can include multiple terms to search for, by both partial match and exact match. It can also contain Boolean operators.

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

The metric schema and `SearchTerm` must be enclosed together in a pair of single quote marks.

- The `Statistic` is the name of any valid CloudWatch statistic. It must be enclosed by single quotes. For more information, see [Statistics \(p. 5\)](#).
- The `Period` is the aggregation time period in seconds.

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

Search Expression Limits

The maximum search expression query size is 1024 characters. You can have as many as five search expressions on one graph. A graph can display as many as 500 time series.

CloudWatch Search Expressions: Tokenization

When you specify a `SearchTerm`, the search function searches for *tokens*, which are substrings that CloudWatch automatically generates from full metric names, dimension names, dimension values, and namespaces. CloudWatch generates tokens distinguished by the camel-case capitalization in the original string. Numeric characters also serve as the start of new tokens, and non-alphanumeric characters serve as delimiters, creating tokens before and after the non-alphanumeric characters.

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

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

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

CloudWatch Search Expressions: Partial Matches

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

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

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

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

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

CloudWatch Search Expressions: Exact Matches

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

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

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

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

CloudWatch Search Expressions: Excluding a Metric Schema

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

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

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

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

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

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

CloudWatch Search Expressions: Specifying Property Names in the Search

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

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

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

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

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

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

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

CloudWatch Search Expressions: Non-Alphanumeric Characters

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

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

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

```
Network_Errors
```

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

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

```
" \ ( )
```

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

CloudWatch Search Expressions: Boolean Operators

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

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

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

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

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

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

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

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

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

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

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

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

CloudWatch Search Expressions: Using Math Expressions

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

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

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

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

CloudWatch Search Expression Examples

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Creating a CloudWatch Graph with a Search Expression

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

To add a graph with a search expression to an existing dashboard

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

2. In the navigation pane, choose **Dashboards** and select a dashboard.
3. Choose **Add widget**.
4. Choose either **Line** or **Stacked area** and choose **Configure**.
5. On the **Graphed metrics** tab, choose **Add a math expression**.
6. For **Details**, enter the search expression that you want. For example, `SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average', 300)`
7. (Optional) To add another search expression or math expression to the graph, choose **Add a math expression**
8. (Optional) After you add a search expression, you can specify a dynamic label to appear on the graph legend for each metric. Dynamic labels display a statistic about the metric and automatically update when the dashboard or graph is refreshed. To add a dynamic label, choose **Graphed metrics** and then **Dynamic labels**.

By default, the dynamic values you add to the label appear at the beginning of the label. You can then click the **Label** value for the metric to edit the label. For more information, see [Using Dynamic Labels \(p. 46\)](#).

9. (Optional) To add a single metric to the graph, choose the **All metrics** tab and drill down to the metric you want.
10. (Optional) To change the time range shown on the graph, choose either **custom** at the top of the graph or one of the time periods to the left of **custom**.
11. (Optional) Horizontal annotations help dashboard users quickly see when a metric has spiked to a certain level or whether the metric is within a predefined range. To add a horizontal annotation, choose **Graph options** and then **Add horizontal annotation**:
 - a. For **Label**, enter a label for the annotation.
 - b. For **Value**, enter the metric value where the horizontal annotation appears.
 - c. For **Fill**, specify whether to use fill shading with this annotation. For example, choose **Above** or **Below** for the corresponding area to be filled. If you specify **Between**, another **Value** field appears, and the area of the graph between the two values is filled.
 - d. For **Axis**, specify whether the numbers in **Value** refer to the metric associated with the left y-axis or the right y-axis if the graph includes multiple metrics.

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

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

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

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

12. (Optional) Vertical annotations help you mark milestones in a graph, such as operational events or the beginning and end of a deployment. To add a vertical annotation, choose **Graph options** and then **Add vertical annotation**:
 - a. For **Label**, enter a label for the annotation. To show only the date and time on the annotation, keep the **Label** field blank.
 - b. For **Date**, specify the date and time where the vertical annotation appears.
 - c. For **Fill**, specify whether to use fill shading before or after a vertical annotation or between two vertical annotations. For example, choose **Before** or **After** for the corresponding area to be filled. If you specify **Between**, another **Date** field appears, and the area of the graph between the two values is filled.

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

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

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

13. Choose **Create widget**.

14. Choose **Save dashboard**.

To use the Metrics view to graph searched metrics

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

Results that match your search appear.

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

or

To refine your search, choose one of the namespaces that appeared in your search results.

5. If you selected a namespace to narrow your results, you can do the following:
 - a. To graph one or more metrics, select the check box next to each metric. To select all metrics, select the check box in the heading row of the table.
 - b. To refine your search, hover over a metric name and choose **Add to search** or **Search for this only**.
 - c. To view help for a metric, select the metric name and choose **What is this?**

The selected metrics appear on the graph.

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

Using Amazon CloudWatch Alarms

You can create both *metric alarms* and *composite alarms* in CloudWatch.

- A *metric alarm* watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. The alarm performs one or more actions based on the value of the metric or expression relative to a threshold over a number of time periods. The action can be an Amazon EC2 action, an Amazon EC2 Auto Scaling action, or a notification sent to an Amazon SNS topic.
- A *composite alarm* includes a rule expression that takes into account the alarm states of other alarms that you have created. The composite alarm goes into ALARM state only if all conditions of the rule are met. The alarms specified in a composite alarm's rule expression can include metric alarms and other composite alarms.

Using composite alarms can reduce alarm noise. You can create multiple metric alarms, and also create a composite alarm and set up alerts only for the composite alarm. For example, a composite might go into ALARM state only when all of the underlying metric alarms are in ALARM state.

Composite alarms can send Amazon SNS notifications when they change state, but cannot perform EC2 actions or Auto Scaling actions.

You can add alarms to CloudWatch dashboards and monitor them visually. When an alarm is on a dashboard, it turns red when it is in the ALARM state, making it easier for you to monitor its status proactively.

Alarms invoke actions for sustained state changes only. CloudWatch alarms don't invoke actions simply because they are in a particular state, the state must have changed and been maintained for a specified number of periods.

After an alarm invokes an action due to a change in state, its subsequent behavior depends on the type of action that you have associated with the alarm. For Amazon EC2 Auto Scaling actions, the alarm continues to invoke the action for every period that the alarm remains in the new state. For Amazon SNS notifications, no additional actions are invoked.

Note

CloudWatch doesn't test or validate the actions that you specify, nor does it detect any Amazon EC2 Auto Scaling or Amazon SNS errors resulting from an attempt to invoke nonexistent actions. Make sure that your alarm actions exist.

Metric Alarm States

A metric alarm has the following possible states:

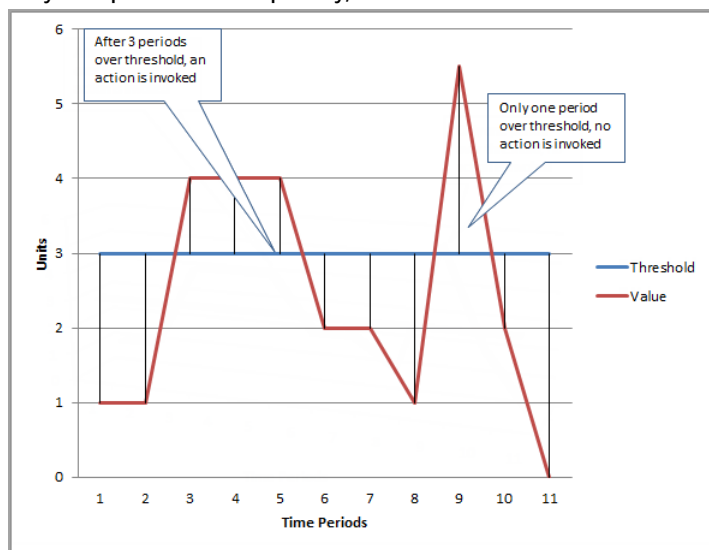
- OK – The metric or expression is within the defined threshold.
- ALARM – The metric or expression is outside of the defined threshold.
- INSUFFICIENT_DATA – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

Evaluating an Alarm

When you create an alarm, you specify three settings to enable CloudWatch to evaluate when to change the alarm state:

- **Period** is the length of time to evaluate the metric or expression to create each individual data point for an alarm. It is expressed in seconds. If you choose one minute as the period, there is one data point every minute.
- **Evaluation Period** is the number of the most recent periods, or data points, to evaluate when determining alarm state.
- **Datapoints to Alarm** is the number of data points within the evaluation period that must be breaching to cause the alarm to go to the `ALARM` state. The breaching data points don't have to be consecutive, they just must all be within the last number of data points equal to **Evaluation Period**.

In the following figure, the alarm threshold for a metric alarm is set to three units. The alarm is configured to go to the `ALARM` state and both **Evaluation Period** and **Datapoints to Alarm** are 3. That is, when all existing data points in the most recent three consecutive periods are above the threshold, the alarm goes to the `ALARM` state. In the figure, this happens in the third through fifth time periods. At period six, the value dips below the threshold, so one of the periods being evaluated is not breaching, and the alarm state changes to `OK`. During the ninth time period, the threshold is breached again, but for only one period. Consequently, the alarm state remains `OK`.



When you configure **Evaluation Period** and **Datapoints to Alarm** as different values, you're setting an "M out of N" alarm. **Datapoints to Alarm** is ("M") and **Evaluation Period** is ("N"). The evaluation interval is the number of data points multiplied by the period. For example, if you configure 4 out of 5 data points with a period of 1 minute, the evaluation interval is 5 minutes. If you configure 3 out of 3 data points with a period of 10 minutes, the evaluation interval is 30 minutes.

Configuring How CloudWatch Alarms Treat Missing Data

Sometimes some data points for a metric with an alarm don't get reported to CloudWatch. For example, this can happen when a connection is lost, a server goes down, or when a metric reports data only intermittently by design.

CloudWatch enables you to specify how to treat missing data points when evaluating an alarm. This can help you configure your alarm to go to the `ALARM` state when appropriate for the type of data being monitored. You can avoid false positives when missing data doesn't indicate a problem.

Similar to how each alarm is always in one of three states, each specific data point reported to CloudWatch falls under one of three categories:

- Not breaching (within the threshold)
- Breaching (violating the threshold)
- Missing

For each alarm, you can specify CloudWatch to treat missing data points as any of the following:

- `notBreaching` – Missing data points are treated as "good" and within the threshold,
- `breaching` – Missing data points are treated as "bad" and breaching the threshold
- `ignore` – The current alarm state is maintained
- `missing` – The alarm doesn't consider missing data points when evaluating whether to change state

The best choice depends on the type of metric. For a metric that continually reports data, such as `CPUUtilization` of an instance, you might want to treat missing data points as `breaching`, because they might indicate that something is wrong. But for a metric that generates data points only when an error occurs, such as `ThrottledRequests` in Amazon DynamoDB, you would want to treat missing data as `notBreaching`. The default behavior is `missing`.

Choosing the best option for your alarm prevents unnecessary and misleading alarm condition changes, and also more accurately indicates the health of your system.

How Alarm State Is Evaluated When Data Is Missing

No matter what value you set for how to treat missing data, when an alarm evaluates whether to change state, CloudWatch attempts to retrieve a higher number of data points than specified by **Evaluation Periods**. The exact number of data points it attempts to retrieve depends on the length of the alarm period and whether it is based on a metric with standard resolution or high resolution. The time frame of the data points that it attempts to retrieve is the *evaluation range*.

Once CloudWatch retrieves these data points, the following happens:

- If no data points in the evaluation range are missing, CloudWatch evaluates the alarm based on the most recent data points collected.
- If some data points in the evaluation range are missing, but the number of existing data points retrieved is equal to or more than the alarm's **Evaluation Periods**, CloudWatch evaluates the alarm state based on the most recent existing data points that were successfully retrieved. In this case, the value you set for how to treat missing data is not needed and is ignored.
- If some data points in the evaluation range are missing, and the number of existing data points that were retrieved is lower than the alarm's number of evaluation periods, CloudWatch fills in the missing data points with the result you specified for how to treat missing data, and then evaluates the alarm. However, any real data points in the evaluation range, no matter when they were reported, are included in the evaluation. CloudWatch uses missing data points only as few times as possible.

In all of these situations, the number of data points evaluated is equal to the value of **Evaluation Periods**. If fewer than the value of **Datapoints to Alarm** are breaching, the alarm state is set to `OK`. Otherwise, the state is set to `ALARM`. The exception is if you have set the alarm to treat missing data points as `missing`. In this case, if CloudWatch could not retrieve enough actual data points during the evaluation range, then the alarm state is set to `INSUFFICIENT_DATA`.

Note

A particular case of this behavior is that CloudWatch alarms might repeatedly re-evaluate the last set of data points for a period of time after the metric has stopped flowing. This re-

evaluation might cause the alarm to change state and re-execute actions, if it had changed state immediately prior to the metric stream stopping. To mitigate this behavior, use shorter periods.

The following tables illustrate examples of the alarm evaluation behavior. In the first table, **Datapoints to Alarm** and **Evaluation Periods** are both 3. CloudWatch retrieves the 5 most recent data points when evaluating the alarm, in case some of the most recent 3 data points are missing. 5 is the evaluation range for the alarm.

Column 2 shows how many of the 3 necessary data points are missing. Even though the most recent 5 data points are evaluated, only 3 (the setting for **Evaluation Periods**) are necessary to evaluate the alarm state. The number of data points in Column 2 is the number of data points that must be "filled in", using the setting for how missing data is being treated.

Columns 3-6 show the alarm state that would be set for each setting of how missing data should be treated, shown at the top of each column. In the data points column, 0 is a non-breaching data point, X is a breaching data point, and - is a missing data point.

Data points	# of missing data points	MISSING	IGNORE	BREACHING	NOT BREACHING
0 - X - X	0	OK	OK	OK	OK
0 - - - -	2	OK	OK	OK	OK
- - - - -	3	INSUFFICIENT - Data in current state	INSUFFICIENT - Data in current state	ALARM	OK
0 X X - X	0	ALARM	ALARM	ALARM	ALARM
- - X - -	2	ALARM	ALARM	ALARM	OK

In the second row of the preceding table, the alarm stays OK even if missing data is treated as breaching, because the one existing data point is not breaching, and this is evaluated along with two missing data points which are treated as breaching. The next time this alarm is evaluated, if the data is still missing it will go to ALARM, as that non-breaching data point will no longer be among the 5 most recent data points retrieved.

In the fourth row, the alarm goes to ALARM state in all cases because there are enough real data points so that the setting for how to treat missing data doesn't need to be considered.

In the final row, the alarm goes to ALARM state because when an alarm's most recent three data points are either breaching or don't exist, the alarm goes to ALARM state.

In the next table, the **Period** is again set to 5 minutes, and **Datapoints to Alarm** is only 2 while **Evaluation Periods** is 3. This is a 2 out of 3, M out of N alarm.

The evaluation range is 5. This is the maximum number of recent data points that are retrieved and can be used in case some data points are missing.

Data points	# of missing data points	MISSING	IGNORE	BREACHING	NOT BREACHING
0 - X - X	0	ALARM	ALARM	ALARM	ALARM
0 0 X 0 X	0	ALARM	ALARM	ALARM	ALARM
0 - X - -	1	OK	OK	ALARM	OK
- - - - 0	2	OK	OK	ALARM	OK

Data points	# of missing data points	MISSING	IGNORE	BREACHING	NOT BREACHING
-- X --	2	ALARM	Retain current state	ALARM	OK

If data points are missing soon after you create an alarm, and the metric was being reported to CloudWatch before you created the alarm, CloudWatch retrieves the most recent data points from before the alarm was created when evaluating the alarm.

High-Resolution Alarms

If you set an alarm on a high-resolution metric, you can specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. There is a higher charge for high-resolution alarms. For more information about high-resolution metrics, see [Publishing Custom Metrics \(p. 50\)](#).

Alarms on Math Expressions

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

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

Alarms based on math expressions can't perform Amazon EC2 actions.

For more information about metric math expressions and syntax, see [Using Metric Math \(p. 52\)](#).

Percentile-Based CloudWatch Alarms and Low Data Samples

When you set a percentile as the statistic for an alarm, you can specify what to do when there is not enough data for a good statistical assessment. You can choose to have the alarm evaluate the statistic anyway and possibly change the alarm state. Or, you can have the alarm ignore the metric while the sample size is low, and wait to evaluate it until there is enough data to be statistically significant.

For percentiles between 0.5 (inclusive) and 1.00 (exclusive), this setting is used when there are fewer than 10/(1-percentile) data points during the evaluation period. For example, this setting would be used if there were fewer than 1000 samples for an alarm on a p99 percentile. For percentiles between 0 and 0.5 (exclusive), the setting is used when there are fewer than 10/percentile data points.

Common Features of CloudWatch Alarms

The following features apply to all CloudWatch alarms:

- You can create up to 5000 alarms per Region per AWS account. To create or update an alarm, you use the CloudWatch console, the [PutMetricAlarm](#) API action, or the [put-metric-alarm](#) command in the AWS CLI.

- Alarm names must contain only ASCII characters.
- You can list any or all of the currently configured alarms, and list any alarms in a particular state by using the CloudWatch console, the [DescribeAlarms](#) API action, or the [describe-alarms](#) command in the AWS CLI.
- You can disable and enable alarms by using the CloudWatch console, the [DisableAlarmActions](#) and [EnableAlarmActions](#) API actions, or the [disable-alarm-actions](#) and [enable-alarm-actions](#) commands in the AWS CLI.
- You can test an alarm by setting it to any state using the [SetAlarmState](#) API action or the [set-alarm-state](#) command in the AWS CLI. This temporary state change lasts only until the next alarm comparison occurs.
- You can create an alarm for a custom metric before you've created that custom metric. For the alarm to be valid, you must include all of the dimensions for the custom metric in addition to the metric namespace and metric name in the alarm definition. To do this, you can use the [PutMetricAlarm](#) API action, or the [put-metric-alarm](#) command in the AWS CLI.
- You can view an alarm's history using the CloudWatch console, the [DescribeAlarmHistory](#) API action, or the [describe-alarm-history](#) command in the AWS CLI. CloudWatch preserves alarm history for two weeks. Each state transition is marked with a unique timestamp. In rare cases, your history might show more than one notification for a state change. The timestamp enables you to confirm unique state changes.
- The number of evaluation periods for an alarm multiplied by the length of each evaluation period can't exceed one day.

Note

Some AWS resources don't send metric data to CloudWatch under certain conditions. For example, Amazon EBS might 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 might notice its state change to `INSUFFICIENT_DATA`. This might indicate that your resource is inactive, and might not necessarily mean that there is a problem. You can specify how each alarm treats missing data. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data](#) (p. 73).

Setting Up Amazon SNS Notifications

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

Note

Alternatively, if you plan to create your CloudWatch alarm using the AWS Management Console, you can skip this procedure because you can create the topic when you create the alarm.

Setting Up an Amazon SNS Topic Using the AWS Management Console

First, create a topic, then subscribe to it. You can optionally publish a test message to the topic.

To create an SNS topic

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. On the Amazon SNS dashboard, under **Common actions**, choose **Create Topic**.
3. In the **Create new topic** dialog box, for **Topic name**, enter a name for the topic (for example, **my-topic**).

4. Choose **Create topic**.
5. Copy the **Topic ARN** for the next task (for example, `arn:aws:sns:us-east-1:111122223333:my-topic`).

To subscribe to an SNS topic

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. In the navigation pane, choose **Subscriptions, Create subscription**.
3. In the **Create subscription** dialog box, for **Topic ARN**, paste the topic ARN that you created in the previous task.
4. For **Protocol**, choose **Email**.
5. For **Endpoint**, enter an email address that you can use to receive the notification, and then choose **Create subscription**.
6. From your email application, open the message from AWS Notifications and confirm your subscription.

Your web browser displays a confirmation response from Amazon SNS.

To publish a test message to an SNS topic

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. In the navigation pane, choose **Topics**.
3. On the **Topics** page, select a topic and choose **Publish to topic**.
4. In the **Publish a message** page, for **Subject**, enter a subject line for your message, and for **Message**, enter a brief message.
5. Choose **Publish Message**.
6. Check your email to confirm that you received the message.

Setting Up an SNS Topic Using the AWS CLI

First, you create an SNS topic, and then you publish a message directly to the topic to test that you have properly configured it.

To set up an SNS topic

1. Create the topic using the `create-topic` command as follows.

```
aws sns create-topic --name my-topic
```

Amazon SNS returns a topic ARN with the following format:

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

2. Subscribe your email address to the topic using the `subscribe` command. If the subscription request succeeds, you receive a confirmation email message.

```
aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:my-topic --protocol
email --notification-endpoint my-email-address
```

Amazon SNS returns the following:

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

3. From your email application, open the message from AWS Notifications and confirm your subscription.

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

4. Check the subscription using the [list-subscriptions-by-topic](#) command.

```
aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-east-1:111122223333:my-
topic
```

Amazon SNS returns the following:

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

5. (Optional) Publish a test message to the topic using the [publish](#) command.

```
aws sns publish --message "Verification" --topic arn:aws:sns:us-east-1:111122223333:my-
topic
```

Amazon SNS returns the following.

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

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

Create a CloudWatch Alarm Based on a Static Threshold

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

CloudWatch has changed the alarm user interface. By default, you're shown the new user interface, but you can choose to return to the old user interface. This topic contains steps for each.

To create an alarm based on a single metric using the new alarms user interface

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

2. In the navigation pane, choose **Alarms, Create Alarm**.
3. Choose **Select Metric** and do one of the following:
 - Choose the service namespace that contains the metric that you want. Continue choosing options as they appear to narrow the choices. When a list of metrics appears, select the check box next to the metric that you want.
 - In the search box, enter the name of a metric, dimension, or resource ID and press Enter. Then choose one of the results and continue until a list of metrics appears. Select the check box next to the metric that you want.
4. Choose the **Graphed metrics** tab.

- a. Under **Statistic**, choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, **p95.45**).
- b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point.

You can also choose whether the y-axis legend appears on the left or right while you're creating the alarm. This preference is used only while you're creating the alarm.

- c. Choose **Select metric**.

The **Specify metric and conditions** page appears, showing a graph and other information about the metric and statistic you have selected.

5. Under **Conditions**, specify the following:
 - a. Enter a name and description for the alarm. The name must contain only ASCII characters.
 - b. For **Whenever *metric* is**, specify whether the metric must be greater than, less than, or equal to the threshold. Under **than...**, specify the threshold value.
 - c. Choose **Additional configuration**. For **Datapoints to alarm**, specify how many evaluation periods (data points) must be in the **ALARM** state to trigger the alarm. If the two values here match, you create an alarm that goes to **ALARM** state if that many consecutive periods are breaching.

To create an M out of N alarm, specify a lower number for the first value than you specify for the second value. For more information, see [Evaluating an Alarm \(p. 72\)](#).
 - d. For **Missing data treatment**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
 - e. If the alarm uses a percentile as the monitored statistic, a **Percentiles with low samples** box appears. Use it to choose whether to evaluate or ignore cases with low sample rates. If you choose **ignore (maintain alarm state)**, the current alarm state is always maintained when the sample size is too low. For more information, see [Percentile-Based CloudWatch Alarms and Low Data Samples \(p. 76\)](#).

6. Choose **Next**.
7. Under **Notification**, select an SNS topic to notify when the alarm is in **ALARM** state, **OK** state, or **INSUFFICIENT_DATA** state.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

To have the alarm not send notifications, choose **Remove**.

8. To have the alarm perform Auto Scaling or EC2 actions, choose the appropriate button and choose the alarm state and action to perform.
9. When finished, choose **Next**.
10. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.

11. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

To use the older user interface to create an alarm based on a single metric

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric** and do one of the following:
 - Choose the service namespace that contains the metric that you want. Continue choosing options as they appear to narrow the choices. When a list of metrics appears, select the check box next to the metric that you want.
 - In the search box, enter the name of a metric, dimension, or resource ID and press Enter. Then choose one of the results and continue until a list of metrics appears. Select the check box next to the metric that you want.
4. Choose the **Graphed metrics** tab.
 - a. Under **Statistic**, choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, **p95.45**).
 - b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point.

You can also choose whether the y-axis legend appears on the left or right while you're creating the alarm. This preference is used only while you're creating the alarm.
 - c. Choose **Select metric**.
5. Enter a name and description for the alarm. The name must contain only ASCII characters.
6. For **Whenever**, specify the alarm condition.
 - a. For **is**, specify whether the metric must be greater than, less than, or equal to the threshold, and specify the threshold value.
 - b. For **for**, specify how many evaluation periods (data points) must be in the **ALARM** state to trigger the alarm. Initially, you can change only the second value, and the first value changes to match your entry. This creates an alarm that goes to **ALARM** state if that many consecutive periods are breaching.

To create an M out of N alarm, choose the pencil icon. You can then change the M number to be different than the N number. For more information, see [Evaluating an Alarm \(p. 72\)](#).
7. Under **Additional settings**, for **Treat missing data as**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
8. If the alarm uses a percentile as the monitored statistic, a **Percentiles with low samples** box appears. Use it to choose whether to evaluate or ignore cases with low sample rates. If you choose **ignore (maintain the alarm state)**, the current alarm state is always maintained when the sample size is too low. For more information, see [Percentile-Based CloudWatch Alarms and Low Data Samples \(p. 76\)](#).
9. Under **Actions**, select the type of action to have the alarm to perform when the alarm is triggered. Use the **+Notification**, **+AutoScaling Action**, and **+EC2 Action** buttons to have the alarm perform multiple actions. Specify at least one action.
10. Choose **Create Alarm**.

You can also add alarms to a dashboard. For more information, see [Add or Remove an Alarm from a CloudWatch Dashboard \(p. 26\)](#).

Creating a CloudWatch Alarm Based on Anomaly Detection

You can create an alarm based on CloudWatch anomaly detection, which mines past metric data and creates a model of expected values. The expected values take into account the typical hourly, daily, and weekly patterns in the metric.

You set a value for the anomaly detection threshold, and CloudWatch uses this threshold with the model to determine the "normal" range of values for the metric. A higher value for the threshold produces a thicker band of "normal" values.

You can choose whether the alarm is triggered when the metric value is above the band of expected values, below the band, or either above or below the band.

For more information, see [Using CloudWatch Anomaly Detection \(p. 141\)](#).

Note

If you create an anomaly detection alarm on a metric that you're already using anomaly detection for in the Metrics console for visualization purposes, the threshold that you set for the alarm doesn't change the threshold that you're already using for visualization. For more information, see [Creating a Graph \(p. 43\)](#).

To create an alarm based on anomaly detection

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric** and do one of the following:
 - Choose the service namespace that contains the metric that you want. To narrow the choices, continue choosing options as they appear. When a list of metrics appears, select the check box next to the metric that you want.
 - In the search box, enter the name of a metric, dimension, or resource ID and press Enter. Then choose one of the results and continue until a list of metrics appears. Select the check box next to the metric that you want.
4. Choose the **Graphed metrics** tab.
 - a. Under **Statistic**, choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, **p95.45**).
 - b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point. For anomaly detection alarms, the value must be one minute or longer.

You can also choose whether the y-axis legend appears on the left or right while you're creating the alarm. This preference is used only while you're creating the alarm.

- c. Choose **Select metric**.

The **Specify metric and conditions** page appears, showing a graph and other information about the metric and statistic you have selected.

5. Under **Conditions**, specify the following:
 - a. Choose **Anomaly detection**.

If the model for this metric and statistic already exists, CloudWatch displays the anomaly detection band in the sample graph at the top of the screen. If the model does not already exist, the model will be generated when you finish creating the alarm. It takes up to 15 minutes for

the actual anomaly detection band generated by the model to appear in the graph. Before that, the band you see is an approximation of the anomaly detection band. To see the graph in a longer time frame, choose **Edit** at the top right of the page.

- b. For **Whenever *metric* is**, specify whether the metric must be greater than, lower than, or outside (in either direction) the band to trigger the alarm.
- c. For **Anomaly detection threshold**, choose the number to use for the anomaly detection threshold. A higher number creates a thicker band of "normal" values that is more tolerant of metric changes, and a lower number creates a thinner band that will go to **ALARM** state with smaller metric deviations. The number does not have to be a whole number.
- d. Choose **Additional configuration**. For **Datapoints to alarm**, specify how many evaluation periods (data points) must be in the **ALARM** state to trigger the alarm. If the two values here match, you create an alarm that goes to **ALARM** state if that many consecutive periods are breaching.

To create an M out of N alarm, specify a lower number for the first value than you specify for the second value. For more information, see [Evaluating an Alarm \(p. 72\)](#).

- e. For **Missing data treatment**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
 - f. If the alarm uses a percentile as the monitored statistic, a **Percentiles with low samples** box appears. Use it to choose whether to evaluate or ignore cases with low sample rates. If you choose **ignore (maintain alarm state)**, the current alarm state is always maintained when the sample size is too low. For more information, see [Percentile-Based CloudWatch Alarms and Low Data Samples \(p. 76\)](#).
6. Choose **Next**.
 7. Under **Notification**, select an SNS topic to notify when the alarm is in **ALARM** state, **OK** state, or **INSUFFICIENT_DATA** state.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

To have the alarm not send notifications, choose **Remove**.
 8. To have the alarm perform EC2 actions, choose the appropriate button and choose the alarm state and action to perform.
 9. When finished, choose **Next**.
 10. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.
 11. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

Modifying an Anomaly Detection Model

Once you have created an alarm, you can adjust the anomaly detection model. You can exclude certain time periods from being used in the model creation. It is critical that you exclude unusual events such as system outages, deployments, and holidays from the training data. You can also specify whether to adjust the model for daylight savings time changes.

To adjust the anomaly detection model for an alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Choose the name of the alarm. Use the search box to find the alarm if necessary.
4. Choose **View in metrics**.

5. In the lower part of the screen, choose **Edit model**.
6. To exclude a time period from being used to produce the model, choose **Add another time range to exclude from training**. Then select or enter the days and times to exclude from training, and choose **Apply**.
7. If the metric is sensitive to daylight savings time changes, select the appropriate time zone in the **Metric timezone** box.
8. Choose **Update**.

Deleting an Anomaly Detection Model

Using anomaly detection for an alarm accrues AWS charges. If you no longer need an anomaly detection model for an alarm, you should delete the alarm and then the model. If you delete the model without deleting the alarm, the alarm automatically recreates the model.

To delete an alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Choose the name of the alarm.
4. Choose **Actions**, **Delete**.

To delete the anomaly detection model that had been used for an alarm

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

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

4. In the results, select the metric that had the anomaly detection model.
5. Choose the **Graphed metrics** tab.
6. In the lower part of the screen, choose **Edit model** and then **Delete model**.

Creating a CloudWatch Alarm Based on a Metric Math Expression

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

CloudWatch has changed the alarm user interface. By default, you're shown the new user interface, but you can choose to return to the old user interface. This topic contains steps for each.

To create an alarm based on a math expression using the new alarm user interface

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

- Choose the service namespace that contains a specific metric. Continue choosing options as they appear to narrow the choices. When a list of metrics appears, select the check box next to the right metric.
- In the search box, enter the name of a metric, dimension, or resource ID and press Enter. Choose one of the results and continue until a list of metrics appears. Select the check box next to the right metric.

(Optional) To add another metric to use in the math expression, under **All metrics**, choose **All**, find the specific metric, and then select the check box next to it. You can add up to 10 metrics.

4. Choose **Graphed metrics**. For each metric added, do the following:
 - a. Under **Statistic**, choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, **p95 . 45**).
 - b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point.

You can also choose whether the y-axis legend appears on the left or right while you're creating the alarm. This preference is used only while you're creating the alarm.

5. Choose **Add a math expression**. A new row appears for the expression.
6. On the new row, under the **Details** column, enter the math expression and press Enter. For information about the functions and syntax that you can use, see [Metric Math Syntax and Functions \(p. 53\)](#).

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

You can change the value of **Id**. It can include numbers, letters, and underscores, and it must start with a lowercase letter. Changing the value of **Id** to a more meaningful name can also make the alarm graph easier to understand.

7. (Optional) Add more math expressions, using both metrics and the results of other math expressions in the formulas of the new math expressions.
8. When you have the expression to use for the alarm, clear the check boxes to the left of every other expression and every metric on the page. Only the check box next to the expression to use for the alarm should be selected. The expression that you choose for the alarm must produce a single time series and show only one line on the graph. Then choose **Select metric**.

The **Specify metric and conditions** page appears, showing a graph and other information about the math expression that you have selected.

9. For **Whenever *expression* is**, specify whether the expression must be greater than, less than, or equal to the threshold. Under **than...**, specify the threshold value.
10. Choose **Additional configuration**. For **Datapoints to alarm**, specify how many evaluation periods (data points) must be in the **ALARM** state to trigger the alarm. If the two values here match, you create an alarm that goes to **ALARM** state if that many consecutive periods are breaching.

To create an M out of N alarm, specify a lower number for the first value than you specify for the second value. For more information, see [Evaluating an Alarm \(p. 72\)](#).

11. For **Missing data treatment**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
12. Choose **Next**.
13. Under **Notification**, select an SNS topic to notify when the alarm is in **ALARM** state, **OK** state, or **INSUFFICIENT_DATA** state.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

To have the alarm not send notifications, choose **Remove**.

14. To have the alarm perform Auto Scaling or EC2 actions, choose the appropriate button and choose the alarm state and action to perform.
15. When finished, choose **Next**.
16. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.
17. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

You can also add alarms to a dashboard. For more information, see [Add or Remove an Alarm from a CloudWatch Dashboard](#) (p. 26).

To create an alarm based on a math expression using the older user interface

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Choose **Select Metric** and do one of the following:
 - Choose the service namespace that contains a specific metric. Continue choosing options as they appear to narrow the choices. When a list of metrics appear, select the check box next to the right metric.
 - In the search box, enter the name of a metric, dimension, or resource ID and press Enter. Choose one of the results and continue until a list of metrics appear. Select the check box next to the right metric.

(Optional) To add another metric to use in the math expression, under **All metrics**, choose **All**, find the specific metric and then select the check box next to it. You can add up to 10 metrics.

4. Choose **Graphed metrics**. For each metric added, do the following:
 - a. Under **Statistic**, choose one of the statistics or predefined percentiles, or specify a custom percentile (for example, **p95.45**).
 - b. Under **Period**, choose the evaluation period for the alarm. When evaluating the alarm, each period is aggregated into one data point.

You can also choose whether the y-axis legend appears on the left or right while you're creating the alarm. This preference is used only while you're creating the alarm.

5. Choose **Add a math expression**. A new row appears for the expression.
6. On the new row, under the **Details** column, enter the math expression and press Enter. For information about the functions and syntax that you can use, see [Metric Math Syntax and Functions](#) (p. 53).

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

You can change the value of **Id**. It can include numbers, letters, and underscores, and it must start with a lowercase letter. Changing the value of **Id** to a more meaningful name can also make the alarm graph easier to understand.

7. (Optional) Add more math expressions, using both metrics and the results of other math expressions in the formulas of the new math expressions.
8. When you have the expression to use for the alarm, clear the check boxes to the left of every other expression and every metric on the page. Only the check box next to the expression to use for the alarm should be selected. The expression that you choose for the alarm must produce a single time series and show only one line on the graph. Then choose **Select metric**.
9. Enter a name and description for the alarm. The name must contain only ASCII characters.

10. For **Whenever**, specify the alarm condition.
 - a. For **is**, specify whether the expression result must be greater than, less than, or equal to the threshold and specify the threshold value.
 - b. For **for**, specify how many evaluation periods (data points) must be in the `ALARM` state to trigger the alarm. Initially, you can change only the second value, and the first value changes to match your entry. This creates an alarm that goes to the `ALARM` state if that many consecutive periods are breaching.

To create an M out of N alarm, choose the pencil icon. You can then change the M number to be different than the N number. For more information, see [Evaluating an Alarm \(p. 72\)](#).
11. Under **Additional settings**, for **Treat missing data as**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
12. Under **Actions**, select the type of action to have the alarm to perform when the alarm is triggered. Choose **+Notification** or **+AutoScaling Action** to have the alarm perform multiple actions. Specify at least one action.
13. Choose **Create Alarm**.

Creating a Composite Alarm

Composite alarms are alarms that determine their alarm state by watching the alarm states of other alarms.

Using composite alarms can help you reduce alarm noise. If you set up a composite alarm to notify you of state changes, but set up the underlying metric alarms to not send notifications themselves, you will be notified only when the alarm state of the composite alarm changes. For example, you could create metric alarms based on both CPU utilization and disk read operations, and specify for these alarms to never take actions. You could then create a composite alarm that goes into `ALARM` state and notifies you only when both of those metric alarms are in `ALARM` state.

In a composite alarm, all underlying alarms must be in the same AWS Region and the same account.

Currently, the only alarm actions that can be taken by composite alarms are notifying SNS topics.

Rule Expressions

Composite alarms include *rule expressions*, which specify which other alarms are to be evaluated to determine the composite alarm's state. For each alarm that you reference in the rule expression, you designate a function that specifies whether that alarm needs to be in `ALARM` state, `OK` state, or `INSUFFICIENT_DATA` state. You can use operators (`AND`, `OR` and `NOT`) to combine multiple functions in a single expression. You can use parenthesis to logically group the functions in your expression.

A rule expression can refer to both metric alarms and other composite alarms.

Functions can include the following:

- `ALARM("alarm-name or alarm-ARN")` is `TRUE` if the named alarm is in `ALARM` state.
- `OK("alarm-name or alarm-ARN")` is `TRUE` if the named alarm is in `OK` state.
- `INSUFFICIENT_DATA("alarm-name or alarm-ARN")` is `TRUE` if the named alarm is in `INSUFFICIENT_DATA` state.
- `TRUE` always evaluates to `TRUE`.
- `FALSE` always evaluates to `FALSE`.

`TRUE` and `FALSE` are useful for testing a complex `AlarmRule` structure, and for testing your alarm actions.

The following are some examples of AlarmRule:

- `ALARM(CPUUtilizationTooHigh) AND ALARM(DiskReadOpsTooHigh)` specifies that the composite alarm goes into ALARM state only if both CPUUtilizationTooHigh and DiskReadOpsTooHigh alarms are in ALARM state.
- `ALARM(CPUUtilizationTooHigh) AND NOT ALARM(DeploymentInProgress)` specifies that the alarm goes to ALARM state if CPUUtilizationTooHigh is in ALARM state and DeploymentInProgress is not in ALARM state. This example reduces alarm noise during a known deployment window.
- `(ALARM(CPUUtilizationTooHigh) OR ALARM(DiskReadOpsTooHigh)) AND OK(NetworkOutTooHigh)` goes into ALARM state if CPUUtilizationTooHigh OR DiskReadOpsTooHigh is in ALARM state, and if NetworkOutTooHigh is in OK state. This provides another example of using a composite alarm to prevent noise. This rule ensures that you are not notified with an alarm action on high CPU or disk usage if a known network problem is also occurring.

The AlarmRule can specify as many as 100 "children" alarms. The AlarmRule expression can have as many as 500 elements. Elements are child alarms, TRUE or FALSE statements, and parentheses.

To create a composite alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. In the list of alarms, select the check boxes next to each of the existing alarms that you want to reference in your new composite alarm. Then choose **Create composite alarm**.
4. In the **Conditions** box, specify the rule expression that the composite alarm will use. Initially, the alarms you selected are listed, joined by the OR logical operator. Each of these alarms has the ALARM state specified.

You can modify the alarm conditions for the composite alarm that you are creating:

- a. For each underlying alarm listed, you can change the required state from ALARM to OK or INSUFFICIENT_DATA.
- b. You can change each OR operator to AND or NOT. You can also add parentheses to group the logical operators.
- c. You can add more alarms to the composite alarm conditions. You can also delete alarms currently listed in the **Conditions** box.

For example, you could specify the following conditions to create a composite alarm that goes into ALARM state if CPUUtilizationTooHigh OR DiskReadOpsTooHigh is in ALARM state, at the same time that NetworkOutTooHigh is in OK state.

```
(ALARM("CPUUtilizationTooHigh") OR  
ALARM("DiskReadOpsTooHigh")) AND  
OK("NetworkOutTooHigh")
```

5. When you are satisfied with the alarm conditions, choose **Next**.
6. Under **Notification**, select an SNS topic to notify when the composite alarm changes state.

To have the alarm not send notifications or take any actions at all, choose **Remove**.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

7. When finished, choose **Next**.
8. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.

9. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

Note

It is possible to create a loop or cycle of composite alarms, where composite alarm A depends on composite alarm B, and composite alarm B also depends on composite alarm A. In this scenario, you can't delete any composite alarm that is part of the cycle because there is always still a composite alarm that depends on that alarm that you want to delete.

To get out of such a situation, you must break the cycle by changing the rule of one of the composite alarms in the cycle to remove a dependency that creates the cycle. The simplest change to make to break a cycle is to change the `AlarmRule` of one of the alarms to `False`. Additionally, the evaluation of composite alarms stops if CloudWatch detects a cycle in the evaluation path.

Editing or Deleting a CloudWatch Alarm

You can edit or delete an existing alarm.

CloudWatch has updated the alarm user interface. By default, you're shown the new user interface, but you can choose to return to the old user interface. This topic contains steps for each.

To edit an alarm using the new alarm user interface

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Choose the name of the alarm.
4. Choose **Edit**.

The **Specify metric and conditions** page appears, showing a graph and other information about the metric and statistic that you selected.

5. To change the metric, choose **Edit**, choose the **All metrics** tab, and do one of the following:
 - Choose the service namespace that contains the metric that you want. Continue choosing options as they appear to narrow the choices. When a list of metrics appears, select the check box next to the metric that you want.
 - In the search box, enter the name of a metric, dimension, or resource ID and press Enter. Then choose one of the results and continue until a list of metrics appears. Select the check box next to the metric that you want.

Choose **Select metric**.

6. To change other aspects of the alarm, choose the appropriate options. To change how many data points must be breaching for the alarm to go into `ALARM` state or to change how missing data is treated, choose **Additional configuration**.
7. Choose **Next**.
8. Under **Notification**, **Auto Scaling action**, and **EC2 action**, optionally edit the actions taken when the alarm is triggered. Then choose **Next**.
9. Optionally change the alarm description.

You can't change the name of an existing alarm. You can copy an alarm and give the new alarm a different name. To copy an alarm, select the check box next to the alarm name in the alarm list and choose **Action**, **Copy**.

10. Choose **Next**.
11. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Update alarm**.

To edit an alarm using the older alarms user interface

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Select the alarm and choose **Actions, Modify**.
4. On the **Modify Alarm** page, update the alarm as necessary and choose **Save Changes**. To modify the expression, metrics, period, or statistic, first choose **Edit** near the top of the screen.

You can't change the name of an existing alarm. You can change the description. Or you can copy the alarm and give the new alarm a different name. To copy an alarm, select the alarm and then choose **Actions, Copy**.

To update an email notification list that was created using the Amazon SNS console

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v3/home>.
2. In the navigation pane, choose **Topics** and then select the ARN for your notification list (topic).
3. Do one of the following:
 - To add an email address, choose **Create subscription**. For **Protocol**, choose **Email**. For **Endpoint**, enter the email address of the new recipient. Choose **Create subscription**.
 - To remove an email address, choose the **Subscription ID**. Choose **Other subscription actions, Delete subscriptions**.
4. Choose **Publish to topic**.

To delete an alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**.
3. Select the check box to the left of the name of the alarm, and choose **Actions, Delete**.
4. Choose **Delete**.

Creating a CPU Usage Alarm That Sends Email

You can create an CloudWatch alarm that sends an email message using Amazon SNS when the alarm changes state from OK to ALARM.

The alarm changes to the ALARM state when the average CPU use of an EC2 instance exceeds a specified threshold for consecutive specified periods.

Setting Up a CPU Usage Alarm Using the AWS Management Console

Use these steps to use the AWS Management Console to create a CPU usage alarm.

To create an alarm based on CPU usage

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms, Create Alarm**.
3. Choose **Select metric**.
4. In the **All metrics** tab, choose **EC2 metrics**.

5. Choose a metric category (for example, **Per-Instance Metrics**).
6. Find the row with the instance that you want listed in the **InstanceId** column and **CPUUtilization** in the **Metric Name** column. Select the check box next to this row, and choose **Select metric**.
7. Under **Specify metric and conditions**, for **Statistic** choose **Average**, choose one of the predefined percentiles, or specify a custom percentile (for example, **p95.45**).
8. Choose a period (for example, **5 minutes**).
9. Under **Conditions**, specify the following:
 - a. For **Threshold type**, choose **Static**.
 - b. For **Whenever CPUUtilization is**, specify **Greater**. Under **than...**, specify the threshold that is to trigger the alarm to go to ALARM state if the CPU utilization exceeds this percentage. For example, 70.
 - c. Choose **Additional configuration**. For **Datapoints to alarm**, specify how many evaluation periods (data points) must be in the ALARM state to trigger the alarm. If the two values here match, you create an alarm that goes to ALARM state if that many consecutive periods are breaching.

To create an M out of N alarm, specify a lower number for the first value than you specify for the second value. For more information, see [Evaluating an Alarm \(p. 72\)](#).
 - d. For **Missing data treatment**, choose how to have the alarm behave when some data points are missing. For more information, see [Configuring How CloudWatch Alarms Treat Missing Data \(p. 73\)](#).
 - e. If the alarm uses a percentile as the monitored statistic, a **Percentiles with low samples** box appears. Use it to choose whether to evaluate or ignore cases with low sample rates. If you choose **ignore (maintain alarm state)**, the current alarm state is always maintained when the sample size is too low. For more information, see [Percentile-Based CloudWatch Alarms and Low Data Samples \(p. 76\)](#).
10. Choose **Next**.
11. Under **Notification**, choose **In alarm** and select an SNS topic to notify when the alarm is in ALARM state

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

To have the alarm not send notifications, choose **Remove**.
12. When finished, choose **Next**.
13. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.
14. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

Setting Up a CPU Usage Alarm Using the AWS CLI

Use these steps to use the AWS CLI to create a CPU usage alarm.

To create an alarm that sends email based on CPU usage

1. Set up an SNS topic. For more information, see [Setting Up Amazon SNS Notifications \(p. 77\)](#).
2. Create an alarm using the `put-metric-alarm` command as follows.

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
when CPU exceeds 70%" --metric-name CPUUtilization --namespace AWS/EC2 --statistic
Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --
```

```
dimensions Name=InstanceId,Value=i-12345678 --evaluation-periods 2 --alarm-actions  
arn:aws:sns:us-east-1:111122223333:my-topic --unit Percent
```

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

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

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

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

- c. Check that you have received an email notification about the alarm.

Creating a Load Balancer Latency Alarm That Sends Email

You can set up an Amazon SNS notification and configure an alarm that monitors latency exceeding 100 ms for your Classic Load Balancer.

Setting Up a Latency Alarm Using the AWS Management Console

Use these steps to use the AWS Management Console to create a load balancer latency alarm.

To create a load balancer latency alarm that sends email

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms, Create Alarm**.
3. Under **CloudWatch Metrics by Category**, choose the **ELB Metrics** category.
4. Select the row with the Classic Load Balancer and the **Latency** metric.
5. For the statistic, choose **Average**, choose one of the predefined percentiles, or specify a custom percentile (for example, **p95.45**).
6. For the period, choose **1 Minute**.
7. Choose **Next**.
8. Under **Alarm Threshold**, enter a unique name for the alarm (for example, **myHighCpuAlarm**) and a description of the alarm (for example, **Alarm when Latency exceeds 100s**). Alarm names must contain only ASCII characters.
9. Under **Whenever**, for **is**, choose **>** and enter **0.1**. For **for**, enter **3**.
10. Under **Additional settings**, for **Treat missing data as**, choose **ignore (maintain alarm state)** so that missing data points don't trigger alarm state changes.

For **Percentiles with low samples**, choose **ignore (maintain the alarm state)** so that the alarm evaluates only situations with adequate numbers of data samples.

11. Under **Actions**, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose an existing SNS topic or create a new one.

To create an SNS topic, choose **New list**. For **Send notification to**, enter a name for the SNS topic (for example, **myHighCpuAlarm**), and for **Email list**, enter a comma-separated list of email

addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent.

12. Choose **Create Alarm**.

Setting Up a Latency Alarm Using the AWS CLI

Use these steps to use the AWS CLI to create a load balancer latency alarm.

To create a load balancer latency alarm that sends email

1. Set up an SNS topic. For more information, see [Setting Up Amazon SNS Notifications \(p. 77\)](#).
2. Create the alarm using the `put-metric-alarm` command as follows:

```
aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm when Latency exceeds 100s" --metric-name Latency --namespace AWS/ELB --statistic Average --period 60 --threshold 100 --comparison-operator GreaterThanThreshold --dimensions Name=LoadBalancerName,Value=my-server --evaluation-periods 3 --alarm-actions arn:aws:sns:us-east-1:111122223333:my-topic --unit Seconds
```

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

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value OK
```

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

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason "initializing" --state-value ALARM
```

- c. Check that you have received an email notification about the alarm.

Creating a Storage Throughput Alarm that Sends Email

You can set up an SNS notification and configure an alarm that sends email when Amazon EBS exceeds 100 MB throughput.

Setting Up a Storage Throughput Alarm Using the AWS Management Console

Use these steps to use the AWS Management Console to create an alarm based on Amazon EBS throughput.

To create a storage throughput alarm that sends email

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. Under **EBS Metrics**, choose a metric category.

4. Select the row with the volume and the **VolumeWriteBytes** metric.
5. For the statistic, choose **Average**. For the period, choose **5 Minutes**. Choose **Next**.
6. Under **Alarm Threshold**, enter a unique name for the alarm (for example, **myHighWriteAlarm**) and a description of the alarm (for example, **VolumeWriteBytes exceeds 100,000 KiB/s**). Alarm names must contain only ASCII characters.
7. Under **Whenever**, for **is**, choose **>** and enter **100000**. For **for**, enter **15** consecutive periods.

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

8. Under **Additional settings**, for **Treat missing data as**, choose **ignore (maintain alarm state)** so that missing data points don't trigger alarm state changes.
9. Under **Actions**, for **Whenever this alarm**, choose **State is ALARM**. For **Send notification to**, choose an existing SNS topic or create one.

To create an SNS topic, choose **New list**. For **Send notification to**, enter a name for the SNS topic (for example, **myHighCpuAlarm**), and for **Email list**, enter a comma-separated list of email addresses to be notified when the alarm changes to the ALARM state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

10. Choose **Create Alarm**.

Setting Up a Storage Throughput Alarm Using the AWS CLI

Use these steps to use the AWS CLI to create an alarm based on Amazon EBS throughput.

To create a storage throughput alarm that sends email

1. Create an SNS topic. For more information, see [Setting Up Amazon SNS Notifications \(p. 77\)](#).
2. Create the alarm.

```
aws cloudwatch put-metric-alarm --alarm-name ebs-mon --alarm-description "Alarm when EBS volume exceeds 100MB throughput" --metric-name VolumeReadBytes --namespace AWS/EBS --statistic Average --period 300 --threshold 100000000 --comparison-operator GreaterThanThreshold --dimensions Name=VolumeId,Value=my-volume-id --evaluation-periods 3 --alarm-actions arn:aws:sns:us-east-1:111122223333:my-alarm-topic --insufficient-data-actions arn:aws:sns:us-east-1:111122223333:my-insufficient-data-topic
```

3. Test the alarm by forcing an alarm state change using the [set-alarm-state](#) command.

- a. Change the alarm state from `INSUFFICIENT_DATA` to `OK`.

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing" --state-value OK
```

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

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing" --state-value ALARM
```

- c. Change the alarm state from `ALARM` to `INSUFFICIENT_DATA`.

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason "initializing" --state-value INSUFFICIENT_DATA
```

- d. Check that you have received an email notification about the alarm.

Create Alarms to Stop, Terminate, Reboot, or Recover an Instance

Using Amazon CloudWatch alarm actions, you can create alarms that automatically stop, terminate, reboot, or recover your EC2 instances. You can use the stop or terminate actions to help you save money when you no longer need an instance to be running. You can use the reboot and recover actions to automatically reboot those instances or recover them onto new hardware if a system impairment occurs.

There are a number of scenarios in which you might want to automatically stop or terminate your instance. For example, you might have instances dedicated to batch payroll processing jobs or scientific computing tasks that run for a period of time and then complete their work. Rather than letting those instances sit idle (and accrue charges), you can stop or terminate them, which helps you to save money. The main difference between using the stop and the terminate alarm actions is that you can easily restart a stopped instance if you need to run it again later. You can also keep the same instance ID and root volume. However, you cannot restart a terminated instance. Instead, you must launch a new instance.

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

To set up a CloudWatch alarm action that can reboot, stop, or terminate an instance, you must use a service-linked IAM role, *AWSServiceRoleForCloudWatchEvents*. The *AWSServiceRoleForCloudWatchEvents* IAM role enables AWS to perform alarm actions on your behalf.

To create the service-linked role for CloudWatch Events, use the following command:

```
aws iam create-service-linked-role --aws-service-name events.amazonaws.com
```

Console Support

You can create alarms using the CloudWatch console or the Amazon EC2 console. The procedures in this documentation use the CloudWatch console. For procedures that use the Amazon EC2 console, see [Create Alarms That Stop, Terminate, Reboot, or Recover an Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Permissions

If you are using an AWS Identity and Access Management (IAM) account to create or modify an alarm, you must have the following permissions:

- `iam:CreateServiceLinkedRole`, `iam:GetPolicy`, `iam:GetPolicyVersion`, and `iam:GetRole` — For all alarms with Amazon EC2 actions
- `ec2:DescribeInstanceStatus` and `ec2:DescribeInstances` — For all alarms on Amazon EC2 instance status metrics
- `ec2:StopInstances` — For alarms with stop actions
- `ec2:TerminateInstances` — For alarms with terminate actions
- No specific permissions are needed for alarms with recover actions.

If you have read/write permissions for Amazon CloudWatch but not for Amazon EC2, you can still create an alarm but the stop or terminate actions aren't performed on the instance. However, if you are later granted permission to use the associated Amazon EC2 API actions, the alarm actions you created earlier are performed. For more information, see [Permissions and Policies](#) in the *IAM User Guide*.

If you want to use an IAM role to stop, terminate, or reboot an instance using an alarm action, you can only use the `AWSServiceRoleForCloudWatchEvents` role. Other IAM roles are not supported. However, you can still see the alarm state and perform any other actions such as Amazon SNS notifications or Amazon EC2 Auto Scaling policies.

Contents

- [Adding Stop Actions to Amazon CloudWatch Alarms \(p. 96\)](#)
- [Adding Terminate Actions to Amazon CloudWatch Alarms \(p. 97\)](#)
- [Adding Reboot Actions to Amazon CloudWatch Alarms \(p. 97\)](#)
- [Adding Recover Actions to Amazon CloudWatch Alarms \(p. 98\)](#)
- [Viewing the History of Triggered Alarms and Actions \(p. 100\)](#)

Adding Stop Actions to Amazon CloudWatch Alarms

You can create an alarm that stops an Amazon EC2 instance when a certain threshold has been met. For example, you may run development or test instances and occasionally forget to shut them off. You can create an alarm that is triggered when the average CPU utilization percentage has been lower than 10 percent for 24 hours, signaling that it is idle and no longer in use. You can adjust the threshold, duration, and period to suit your needs, plus you can add an SNS notification, so that you will receive an email when the alarm is triggered.

Amazon EC2 instances that use an Amazon Elastic Block Store volume as the root device can be stopped or terminated, whereas instances that use the instance store as the root device can only be terminated.

To create an alarm to stop an idle instance using the Amazon CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. For the **Select Metric** step, do the following:
 - a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.
 - b. Select the row with the instance and the **CPUUtilization** metric.
 - c. For the statistic, choose **Average**.
 - d. Choose a period (for example, **1 Hour**).
 - e. Choose **Next**.
4. For the **Define Alarm** step, do the following:
 - a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Stop EC2 instance) and a description of the alarm (for example, Stop EC2 instance when CPU is idle too long). Alarm names must contain only ASCII characters.
 - b. Under **Whenever**, for **is**, choose **<** and type **10**. For **for**, type **24** consecutive periods.

A graphical representation of the threshold is shown under **Alarm Preview**.
 - c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Stop_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.
 - d. Choose **EC2 Action**.
 - e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Stop this instance**.

- f. Choose **Create Alarm**.

Adding Terminate Actions to Amazon CloudWatch Alarms

You can create an alarm that terminates an EC2 instance automatically when a certain threshold has been met (as long as termination protection is not enabled for the instance). For example, you might want to terminate an instance when it has completed its work, and you don't need the instance again. If you might want to use the instance later, you should stop the instance instead of terminating it. For information about enabling and disabling termination protection for an instance, see [Enabling Termination Protection for an Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

To create an alarm to terminate an idle instance using the Amazon CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms**, **Create Alarm**.
3. For the **Select Metric** step, do the following:
 - a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.
 - b. Select the row with the instance and the **CPUUtilization** metric.
 - c. For the statistic, choose **Average**.
 - d. Choose a period (for example, **1 Hour**).
 - e. Choose **Next**.
4. For the **Define Alarm** step, do the following:
 - a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Terminate EC2 instance) and a description of the alarm (for example, Terminate EC2 instance when CPU is idle for too long). Alarm names must contain only ASCII characters.
 - b. Under **Whenever**, for **is**, choose **<** and type **10**. For **for**, type **24** consecutive periods.

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

- c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Terminate_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

- d. Choose **EC2 Action**.
- e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Terminate this instance**.
- f. Choose **Create Alarm**.

Adding Reboot Actions to Amazon CloudWatch Alarms

You can create an Amazon CloudWatch alarm that monitors an Amazon EC2 instance and automatically reboots the instance. The reboot alarm action is recommended for Instance Health Check failures (as opposed to the recover alarm action, which is suited for System Health Check failures). An instance reboot is equivalent to an operating system reboot. In most cases, it takes only a few minutes to reboot

your instance. When you reboot an instance, it remains on the same physical host, so your instance keeps its public DNS name, private IP address, and any data on its instance store volumes.

Rebooting an instance doesn't start a new instance billing hour, unlike stopping and restarting your instance. For more information about rebooting an instance, see [Reboot Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Important

To avoid a race condition between the reboot and recover actions, avoid setting the same evaluation period for both a reboot alarm and a recover alarm. We recommend that you set reboot alarms to three evaluation periods of one minute each.

To create an alarm to reboot an instance using the Amazon CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms, Create Alarm**.
3. For the **Select Metric** step, do the following:
 - a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.
 - b. Select the row with the instance and the **StatusCheckFailed_Instance** metric.
 - c. For the statistic, choose **Minimum**.
 - d. Choose a period (for example, **1 Minute**) and choose **Next**.
4. For the **Define Alarm** step, do the following:
 - a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Reboot EC2 instance) and a description of the alarm (for example, Reboot EC2 instance when health checks fail). Alarm names must contain only ASCII characters.
 - b. Under **Whenever**, for **is**, choose **>** and type **0**. For **for**, type **3** consecutive periods.

A graphical representation of the threshold is shown under **Alarm Preview**.
 - c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Reboot_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the **ALARM** state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.
 - d. Choose **EC2 Action**.
 - e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Reboot this instance**.
 - f. Choose **Create Alarm**.

Adding Recover Actions to Amazon CloudWatch Alarms

You can create an Amazon CloudWatch alarm that monitors an Amazon EC2 instance and automatically recovers the instance if it becomes impaired due to an underlying hardware failure or a problem that requires AWS involvement to repair. Terminated instances cannot be recovered. A recovered instance is identical to the original instance, including the instance ID, private IP addresses, Elastic IP addresses, and all instance metadata.

When the `StatusCheckFailed_System` alarm is triggered, and the recover action is initiated, you will be notified by the Amazon SNS topic that you chose when you created the alarm and associated

the recover action. During instance recovery, the instance is migrated during an instance reboot, and any data that is in-memory is lost. When the process is complete, information is published to the SNS topic you've configured for the alarm. Anyone who is subscribed to this SNS topic will receive an email notification that includes the status of the recovery attempt and any further instructions. You will notice an instance reboot on the recovered instance.

The recover action can be used only with `StatusCheckFailed_System`, not with `StatusCheckFailed_Instance`.

Examples of problems that cause system status checks to fail include:

- Loss of network connectivity
- Loss of system power
- Software issues on the physical host
- Hardware issues on the physical host that impact network reachability

The recover action is supported only on:

- The A1, C3, C4, C5, C5n, M3, M4, M5, M5a, P3, R3, R4, R5, R5a, T2, T3, X1, and X1e instance types
- Instances in a VPC
- Instances with default or dedicated instance tenancy
- Instances that use Amazon EBS volumes only (do not configure instance store volumes)

If your instance has a public IPv4 address, it retains the public IP address after recovery.

Important

To avoid a race condition between the reboot and recover actions, avoid setting the same evaluation period for both a reboot alarm and a recover alarm. We recommend that you set recover alarms to two evaluation periods of one minute each and reboot alarms to three evaluation periods of one minute each.

To create an alarm to recover an instance using the Amazon CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
 2. In the navigation pane, choose **Alarms, Create Alarm**.
 3. For the **Select Metric** step, do the following:
 - a. Under **EC2 Metrics**, choose **Per-Instance Metrics**.
 - b. Select the row with the instance and the **StatusCheckFailed_System** metric.
 - c. For the statistic, choose **Minimum**.
 - d. Choose a period (for example, **1 Minute**).
- Important**
- To avoid a race condition between the reboot and recover actions, avoid setting the same evaluation period for both a reboot alarm and a recover alarm. We recommend that you set recover alarms to two evaluation periods of one minute each.
- e. Choose **Next**.
4. For the **Define Alarm** step, do the following:
 - a. Under **Alarm Threshold**, type a unique name for the alarm (for example, Recover EC2 instance) and a description of the alarm (for example, Recover EC2 instance when health checks fail). Alarm names must contain only ASCII characters.
 - b. Under **Whenever**, for **is**, choose **>** and type **0**. For **for**, type **2** consecutive periods.
 - c. Under **Notification**, for **Send notification to**, choose an existing SNS topic or create a new one.

To create an SNS topic, choose **New list**. For **Send notification to**, type a name for the SNS topic (for example, Recover_EC2_Instance). For **Email list**, type a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state. Each email address is sent a topic subscription confirmation email. You must confirm the subscription before notifications can be sent to an email address.

- d. Choose **EC2 Action**.
- e. For **Whenever this alarm**, choose **State is ALARM**. For **Take this action**, choose **Recover this instance**.
- f. Choose **Create Alarm**.

Viewing the History of Triggered Alarms and Actions

You can view alarm and action history in the Amazon CloudWatch console. Amazon CloudWatch keeps the last two weeks' worth of alarm and action history.

To view the history of triggered alarms and actions

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms** and select an alarm.
3. To view the most recent state transition along with the time and metric values, choose **Details**.
4. To view the most recent history entries, choose **History**.

Creating a Billing Alarm to Monitor Your Estimated AWS Charges

You can monitor your estimated AWS charges by using Amazon CloudWatch. When you enable the monitoring of estimated charges for your AWS account, the estimated charges are calculated and sent several times daily to CloudWatch as metric data.

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

The alarm triggers when your account billing exceeds the threshold you specify. It triggers only when actual billing exceeds the threshold. It doesn't use projections based on your usage so far in the month.

If you create a billing alarm at a time when your charges have already exceeded the threshold, the alarm goes to the `ALARM` state immediately.

Tasks

- [Enabling Billing Alerts \(p. 100\)](#)
- [Creating a Billing Alarm \(p. 101\)](#)
- [Deleting a Billing Alarm \(p. 102\)](#)

Enabling Billing Alerts

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

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

Requirements

- You must be signed in using account root user credentials or as an IAM user that has been given permission to view billing information.
- For consolidated billing accounts, billing data for each linked account can be found by logging in as the paying account. You can view billing data for total estimated charges and estimated charges by service for each linked account, in addition to the consolidated account.
- In a consolidated billing account, member linked account metrics are captured only if the payer account enables the **Receive Billing Alerts** preference. If you change which account is your master/payer account, you must enable the billing alerts in the new master/payer account.
- The account must not be part of the Amazon Partner Network (APN) because billing metrics are not published to CloudWatch for APN accounts. For more information, see [AWS Partner Network](#).

To enable the monitoring of estimated charges

1. Open the Billing and Cost Management console at <https://console.aws.amazon.com/billing/>.
2. In the navigation pane, choose **Billing Preferences**.
3. Choose **Receive Billing Alerts**.
4. Choose **Save preferences**.

Creating a Billing Alarm

Important

Before you can create a billing alarm, you must enable billing alerts in your account, or in the master/payer account if you are using consolidated billing. For more information, see [Enabling Billing Alerts \(p. 100\)](#).

In this procedure, you create an alarm that sends an email message when your estimated charges for AWS exceed a specified threshold. This procedure uses the advanced options. For more information about using the simple options, see [Create a Billing Alarm \(p. 413\)](#) in *Monitor Your Estimated Charges Using CloudWatch*.

To create a billing alarm using the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the Region to US East (N. Virginia). Billing metric data is stored in this Region and represents worldwide charges.
3. In the navigation pane, choose **Alarms, Create Alarm**.
4. Choose **Select metric**. In the **All metrics** tab, choose **Billing, Total Estimated Charge**.
5. Select the check box next to **EstimatedCharges**, and choose **Select metric**.
6. Under **Conditions**, choose **Static**.
7. For **Whenever EstimatedCharges is**, choose **Greater**.
8. For **than**, enter the monthly amount (for example, **200**) that must be exceeded to trigger the alarm.

Note

The preview graph displays your current charges for the month.

9. Choose **Next**.
10. For **Select an SNS topic**, select an SNS topic to notify when the alarm is in ALARM state, or create a new topic to be notified.

To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**.

11. When finished, choose **Next**.
12. Enter a name and description for the alarm. The name must contain only ASCII characters. Then choose **Next**.
13. Under **Preview and create**, confirm that the information and conditions are what you want, then choose **Create alarm**.

Deleting a Billing Alarm

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

To delete a billing alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the Region to US East (N. Virginia). Billing metric data is stored in this Region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**.
4. Select the check box next to the alarm and choose **Actions, Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Hiding Amazon EC2 Auto Scaling Alarms

When you view your alarms in the AWS Management Console, you can hide the alarms related to Amazon EC2 Auto Scaling. This feature is available only in the AWS Management Console.

To temporarily hide Amazon EC2 Auto Scaling alarms

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms** and select **Hide all AutoScaling alarms**.

Using Synthetic Monitoring

You can use Amazon CloudWatch Synthetics to create *canaries*, configurable scripts that run on a schedule, to monitor your endpoints and APIs. Canaries follow the same routes and perform the same actions as a customer, which makes it possible for you to continually verify your customer experience even when you don't have any customer traffic on your applications. By using canaries, you can discover issues before your customers do.

Canaries are Node.js scripts. They create Lambda functions in your account that use Node.js as a framework. Canaries can use the Puppeteer Node.js library to perform functions on your applications. Canaries work over HTTP and HTTPS protocols.

Canaries check the availability and latency of your endpoints and can store load time data and screenshots of the UI. They monitor your REST APIs, URLs, and website content, and they can check for unauthorized changes from phishing, code injection and cross-site scripting.

For a video demonstration of canaries, see the [Amazon CloudWatch Synthetics Demo](#) video.

You can run a canary once or on a regular schedule. Scheduled canaries can run 24 hours a day, as often as once per minute.

For information about security issues to consider before you create and run canaries, see [Security Considerations for Synthetics Canaries \(p. 450\)](#).

By default, canaries create CloudWatch metrics in the `CloudWatchSynthetics` namespace. They create metrics with the names `SuccessPercent`, `Failed`, and `Duration`. These metrics have `CanaryName` as a dimension. Canaries that use the `executeStep()` function from the function library also have `StepName` as a dimension. For more information about the canary function library, see [Library Functions Available for Canary Scripts \(p. 115\)](#).

CloudWatch Synthetics integrates well with CloudWatch ServiceLens, which uses CloudWatch with AWS X-Ray to provide an end-to-end view of your services to help you more efficiently pinpoint performance bottlenecks and identify impacted users. Canaries that you create with CloudWatch Synthetics appear on the ServiceLens service map. For more information about ServiceLens, see [Using ServiceLens to Monitor the Health of Your Applications \(p. 123\)](#).

CloudWatch Synthetics is currently available in the following AWS Regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Stockholm)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Mumbai)
- South America (São Paulo)

Topics

- [Required Roles and Permissions for CloudWatch Canaries \(p. 104\)](#)
- [Creating a Canary \(p. 107\)](#)
- [Viewing Canary Statistics and Details \(p. 121\)](#)
- [Deleting a Canary \(p. 122\)](#)

Required Roles and Permissions for CloudWatch Canaries

To view canary details and the results of canary runs, you must be signed in as an IAM user who has either the `CloudWatchSyntheticsFullAccess` or the `CloudWatchSyntheticsReadOnlyAccess` attached. To read all Synthetics data in the console, you also need the `AmazonS3ReadOnlyAccess` and `CloudWatchReadOnlyAccess` policies. To view the source code used by canaries, you also need the `AWSLambdaReadOnlyAccess` policy.

To create canaries, you must be signed in as an IAM user who has the `CloudWatchSyntheticsFullAccess` policy or a similar set of permissions. To create IAM roles for the canaries, you also need the following inline policy statement:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
        "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
      ]
    }
  ]
}
```

Important

Granting a user the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions gives that user full administrative access to your AWS account. For example, a user with these permissions can create a policy that has full permissions for all resources and can attach that policy to any role. Be very careful about who you grant these permissions to.

For information about attaching policies and granting permissions to users, see [Changing Permissions for an IAM User](#) and [To embed an inline policy for a user or role](#).

CloudWatchSyntheticsFullAccess

Here are the contents of the `CloudWatchSyntheticsFullAccess` policy:

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



```

    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:CreateBucket",
      "s3:PutBucketEncryption",
      "s3:PutEncryptionConfiguration"
    ],
    "Resource": [
      "arn:aws:s3:::cw-syn-results-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:ListRoles",
      "s3:ListAllMyBuckets",
      "s3:GetBucketLocation"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Resource": "arn:aws:s3:::cw-syn-*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::aws-synthetics-library-*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:PassRole"
    ],
    "Resource": [
      "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
    ],
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": [
          "lambda.amazonaws.com",
          "synthetics.amazonaws.com"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:GetRole"
    ],
    "Resource": [
      "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
    ]
  },
  {
    "Effect": "Allow",

```

```

        "Action": [
            "cloudwatch:GetMetricData",
            "cloudwatch:GetMetricStatistics"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:DeleteAlarms"
        ],
        "Resource": [
            "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "cloudwatch:DescribeAlarms"
        ],
        "Resource": [
            "arn:aws:cloudwatch:*:*:alarm:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lambda:CreateFunction",
            "lambda:AddPermission",
            "lambda:PublishVersion",
            "lambda:UpdateFunctionConfiguration"
        ],
        "Resource": [
            "arn:aws:lambda:*:*:function:cwsyn-*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "lambda:GetLayerVersionByArn",
            "lambda:GetLayerVersion",
            "lambda:PublishLayerVersion"
        ],
        "Resource": [
            "arn:aws:lambda:*:*:layer:cwsyn-*",
            "arn:aws:lambda:*:*:layer:Synthetics:*"
        ]
    }
]
}

```

CloudWatchSyntheticsReadOnlyAccess

Here are the contents of the CloudWatchSyntheticsReadOnlyAccess policy:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "synthetics:Describe*",
        "synthetics:Get*"
      ],

```

```
        "Resource": "*"
    }
  ]
}
```

Creating a Canary

Important

Ensure that you use Synthetics canaries to monitor only endpoints and APIs where you have ownership or permissions. Depending on canary run frequency settings, these endpoints might experience increased traffic.

You can use a blueprint provided by CloudWatch to create your canary or you can write your own script. For more information, see [Using Canary Blueprints \(p. 108\)](#).

If you are writing your own script, you can use several functions that CloudWatch Synthetics has built into a library. For more information, see [Canary Runtime Versions \(p. 111\)](#).

To create a canary

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Canaries**.
3. Choose **Create Canary**.
4. Choose one of the following:
 - To base your canary on a blueprint script, choose **Use a blueprint**, and then choose the type of canary you want to create. For more information about what each type of blueprint does, see [Using Canary Blueprints \(p. 108\)](#).
 - To upload your own Node.js script to create a custom canary, choose **Upload a script**.

You can then drag your script into the **Script** area or choose **Browse files** to navigate to the script in your file system.
 - To import your script from an S3 bucket, choose **Import from S3**. Under **Source location**, enter the complete path to your canary or choose **Browse S3**.

You must have `s3:GetObject` and `s3:GetObjectVersion` permissions for the S3 bucket that you use. The bucket must be in the same AWS Region where you are creating the canary.
5. Under **Name**, enter a name for your canary. The name is used on many pages, so we recommend that you give it a descriptive name that distinguishes it from other canaries.
6. Under **Application or endpoint URL**, enter the URL that you want the canary to test. This URL must include the protocol (such as `https://`).

If you want the canary to test an endpoint on a VPC, you must also enter information about your VPC later in this procedure.
7. If you are using your own script for the canary, under **Lambda handler**, enter the entry point where you want the canary to start. The string that you enter must end with `.handler`.
8. Under **Schedule**, choose whether to run this canary just once or run it regularly on a schedule. When you use the CloudWatch console to create a canary, the options for canary frequency are once per minute, once per 5 minutes, and once per hour. If you use the AWS CLI or APIs to create a canary, you can choose other frequency options.
9. Under **Data retention**, specify how long to retain information about both failed and successful canary runs. The range is 1-455 days.
10. Under **Data Storage**, select the S3 bucket to use to store the data from the canary runs. If you leave this blank, a default S3 bucket is used or created.

11. Under **Access permissions**, choose whether to create an IAM role to run the canary or use an existing one.

If you use the CloudWatch console to create a role for a canary when you create the canary, you can't re-use the role for other canaries, because these roles are specific to just one canary. If you have manually created a role that works for multiple canaries, you can use that existing role.

To use an existing role, you must have the `iam:PassRole` permission to pass that role to Synthetics and Lambda. You must also have the `iam:GetRole` permission.

12. (Optional) Under **Alarms**, choose whether you want default CloudWatch alarms to be created for this canary.
13. (Optional) To have this canary test an endpoint that is on a VPC, choose **VPC settings**, and then do the following:
 - a. Select the VPC that hosts the endpoint.
 - b. Select one or more subnets on your VPC. You must select a private subnet because a Lambda instance can't be configured to run in a public subnet when an IP address cannot be assigned to the Lambda instance during execution. For more information, see [Configuring a Lambda Function to Access Resources in a VPC](#).
 - c. Select one or more security groups on your VPC.

If the endpoint is on a VPC, you must enable your canary to send information to CloudWatch and Amazon S3. For more information, see [Running a Canary on a VPC \(p. 120\)](#).

14. (Optional) Under **Tags**, add one or more key-value pairs as tags for this canary. Tags can help you identify and organize your AWS resources and track your AWS costs. For more information, see [Tagging Your Amazon CloudWatch Resources \(p. 417\)](#).

Resources That Are Created for Canaries

When you create a canary, the following resources are created:

- An IAM role with the name `CloudWatchSyntheticsRole-canary-name-uuid` (if you use CloudWatch console to create the canary and specify for a new role to be created for the canary)
- An IAM policy with the name `CloudWatchSyntheticsPolicy-canary-name-uuid`.
- An S3 bucket with the name `cw-syn-results-accountID-region`.
- Alarms with the name `Synthetics-Alarm-MyCanaryName`, if you want alarms to be created for the canary.
- Lambda functions and layers, if you use a blueprint to create the canary. These resources have the prefix `cwsyn-MyCanaryName`.
- CloudWatch Logs log groups with the name `/aws/lambda/cwsyn-MyCanaryName`.

Using Canary Blueprints

This section provides details about each of the canary blueprints and the tasks each blueprint is best suited for. Blueprints are provided for the following canary types:

- Heartbeat monitor
- API Canary
- Broken link checker
- GUI Workflow

When you use a blueprint to create a canary, as you fill out the fields in the CloudWatch console, the **Script editor** area of the page displays the canary you are creating as a Node.js script. You can also edit your canary in this area to customize it further.

Heartbeat Monitoring

Heartbeat scripts load the specified URL and store a screenshot of the page and an HTTP archive file (HAR file). They also store logs of accessed URLs.

You can use the HAR files to view detailed performance data about the web pages. You can analyze the list of web requests and catch performance issues such as time to load for an item.

API Canary

API canaries can test the basic Read and Write functions of a REST API. REST stands for *representational state transfer* and is a set of rules that developers follow when creating an API. One of these rules states that a link to a specific URL should return a piece of data.

In a REST API, each URL is called a *request* and the data sent back is the *response*. Each request consists of the following information:

- The *endpoint*, which is the URL that you request.
- The *method*, which is the type of request that is sent to the server. REST APIs support GET (read), POST (write), PUT (update), PATCH (update), and DELETE (delete) operations.
- The *headers*, which provide information to both the client and the server. They are used for authentication and providing information about the body content. For a list of valid headers, see [HTTP Headers](#).
- The *data* (or *body*), contains information to be sent to the server. This is used only for POST, PUT, PATCH, or DELETE requests.
- The URL that you request.

The API canary blueprint supports GET and POST methods. When you use this blueprint, you must specify headers. For example, you can specify **Authorization** as a **Key** and specify the necessary authorization data as the **Value** for that key.

If you are testing a POST request, you also specify the content to post in the **Data** field.

Method
The HTTP request the canary will be testing

POST ▼

Application or endpoint URL [Info](#)

https:// ▼ www.example.com

Enter the endpoint, API or url that you are testing.

Headers

Key	Value	
Authorization	:1234567890	Remove header
Add header		

Data

```
("type":"my_type","value":500)
```

Broken Link Checker

The broken link checker collects all the links inside the URL that you are testing by using `document.getElementsByTagName('a')`. It tests only up to the number of links that you specify, and the URL itself is counted as the first link. For example, if you want to check all the links on a page that contains five links, you must specify for the canary to follow six links.

The canary detects the following types of link errors:

- 404 Page Not Found
- Invalid Host Name
- Bad URL. For example, the URL is missing a bracket, has extra slashes, or is the wrong protocol.
- Invalid HTTP response code
- The host server returns empty responses with no content and no response code.
- The HTTP requests constantly time out during the canary's run.
- The host consistently drops connections because it is misconfigured or is too busy.

GUI Workflow Builder

The GUI Workflow Builder blueprint verifies that actions can be taken on your web page. For example, if you have a web page with a login form, the canary can populate the user and password fields and submit the form to verify that the web page is working correctly.

When you use a blueprint to create this type of canary, you specify the actions that you want the canary to take on the web page. The actions that you can use are the following:

- **Click**— Selects the element that you specify and simulates a user clicking or choosing the element. To specify the element, use `[id=]` or `a[class=]`.
- **Verify selector**— Verifies that the specified element exists on the web page. This test is useful for verifying that a previous action has caused the correct elements to populate the page. To specify the element to verify, use `[id=]` or `a[class=]`.
- **Verify text**— Verifies that the specified string is contained within the target element. This test is useful for verifying that a previous action has caused the correct text to be displayed. To specify the element, use a format such as `div[@id=]/h1` because this action uses the `waitForXPath` function in Puppeteer.
- **Input text**— Writes the specified text in the target element. To specify the element to verify, use `[id=]` or `a[class=]`.
- **Click with navigation**— Waits for the whole page to load after choosing the specified element. This is most useful when you need to reload the page. To specify the element, use `[id=]` or `a[class=]`.

For example, the following blueprint clicks the **firstButton** on the specified URL, verifies that the expected selector with the expected text appears, inputs the name `Test_Customer` into the **Name** field, clicks the **Login** button, and then verifies that the login is successful by checking for the **Welcome** text on the next page.

Application or endpoint URL [Info](#)

Enter the endpoint, API or url that you are testing.

Workflow builder
Select the actions you would like the canary to take.

Action	Selector	Text	
<input type="button" value="Click"/>	<input type="text" value="[id='firstButton']"/>	<input type="text"/>	<input type="button" value="Remove action"/>
<input type="button" value="Verify selector"/>	<input type="text" value="div[id='screen2Text']"/>	<input type="text"/>	<input type="button" value="Remove action"/>
<input type="button" value="Verify text"/>	<input type="text" value="[@id='screen2Text']/h3"/>	<input type="text" value="Type"/>	<input type="button" value="Remove action"/>
<input type="button" value="Input text"/>	<input type="text" value="input[id='Name']"/>	<input type="text" value="Test_Customer"/>	<input type="button" value="Remove action"/>
<input type="button" value="Click with navigation"/>	<input type="text" value="[id='Login']"/>	<input type="text"/>	<input type="button" value="Remove action"/>
<input type="button" value="Verify text"/>	<input type="text" value="div[@id='welcome']/h1"/>	<input type="text" value="Welcome"/>	<input type="button" value="Remove action"/>

Canary Runtime Versions

When you create or update a canary, you choose a Synthetics runtime version for the canary. A Synthetics runtime is a combination of the Synthetics code that calls your script handler, and the Lambda layers of bundled dependencies.

We recommend that you always use the most recent runtime version for your canaries, to be able to use the latest features and updates made to the Synthetics library.

Synthetics runtime versions are labeled `syn-majorversion .minorversion`. Runtime versions with the same major version number are backward compatible.

The current Synthetics runtime version is `syn-1.0`, which includes the following:

- Synthetics library 1.0
- Synthetics handler code 1.0
- Lambda runtime Node.js 10.x
- Puppeteer-core version 1.14.0
- The Chromium version that matches Puppeteer-core 1.14.0

Be sure that your canary scripts are compatible with Node.js 10.x.

The Lambda code in a canary is configured to have a maximum memory of 1 GB. Each run of a canary times out after a configured timeout value. If no timeout value is specified for a canary, CloudWatch chooses a timeout value based on the canary's frequency.

Canary Runtime Support Policy

Synthetics runtime versions are subject to maintenance and security updates. When any component of a runtime version is no longer supported for security updates, that Synthetics runtime version is deprecated.

You can't create canaries using deprecated runtime versions. Canaries that use deprecated runtimes continue to run. You can stop, start, and delete these canaries. You can update an existing canary that uses a deprecated runtime versions by updating the canary to use a supported runtime version.

Writing a Canary Script

The following sections explain how to write a canary script and how to integrate a canary with other AWS Services.

Topics

- [Creating a Synthetics Canary From Scratch \(p. 112\)](#)
- [Changing an Existing Puppeteer Script to Use as a Synthetics Canary \(p. 113\)](#)
- [Integrating Your Canary with Other AWS Services \(p. 114\)](#)

Creating a Synthetics Canary From Scratch

Here is an example minimal Synthetics Canary script. This script passes as a successful run, and returns a string. To see what a failing canary looks like, change `let fail = false;` to `let fail = true;`.

You must define an entry point function for the canary script. To see how files are uploaded to the Amazon S3 location specified as the canary's `ArtifactsS3Location`, create these files under the `/tmp` folder. After the script runs, the pass/fail status and the duration metrics are published to CloudWatch and the files under `/tmp` are uploaded to S3.

```
const basicCustomEntryPoint = async function () {

    // Insert your code here

    // Perform multi-step pass/fail check

    // Log decisions made and results to /tmp

    // Be sure to wait for all your code paths to complete
    // before returning control back to Synthetics.
    // In that way, your canary will not finish and report success
    // before your code has finished executing

    // Throw to fail, return to succeed
    let fail = false;
    if (fail) {
        throw "Failed basicCanary check.";
    }

    return "Successfully completed basicCanary checks.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

Next, we'll expand the script to use Synthetics logging and make a call using the AWS SDK. For demonstration purposes, this script will create a Amazon DynamoDB client and make a call to the DynamoDB `listTables` API. It logs the response to the request and logs either pass or fail depending on whether the request was successful.

If you have more than a single `.js` file or you have a dependency that your script depends on, you can bundle them all into a single ZIP file that contains the folder structure `nodejs/node_modules/myCanaryFilename.js file and other folders and files.`

Be sure to set your canary's script entry point as `myCanaryFilename.handler` to match the filename of your script's entry point.

```
const log = require('SyntheticsLogger');
const AWS = require('aws-sdk');
// Require any dependencies that your script needs
// Bundle additional files and dependencies into a .zip file with folder structure
// nodejs/node_modules/additional files and folders

const basicCustomEntryPoint = async function () {

  log.info("Starting DynamoDB:listTables canary.");

  let dynamodb = new AWS.DynamoDB();
  var params = {};
  let request = await dynamodb.listTables(params);
  try {
    let response = await request.promise();
    log.info("listTables response: " + JSON.stringify(response));
  } catch (err) {
    log.error("listTables error: " + JSON.stringify(err), err.stack);
    throw err;
  }

  return "Successfully completed DynamoDB:listTables canary.";
};

exports.handler = async () => {
  return await basicCustomEntryPoint();
};
```

Changing an Existing Puppeteer Script to Use as a Synthetics Canary

This section explains how to take Puppeteer scripts and modify them to run as Synthetics canary scripts. For more information about Puppeteer, see [Puppeteer API v1.14.0](#).

We'll start with this example Puppeteer script:

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({path: 'example.png'});

  await browser.close();
})();
```

The conversion steps are as follows:

- Create and export a `handler` function. The handler is the entry point function for the script.

```
const basicPuppeteerExample = async function () {};
```

```
exports.handler = async () => {
  return await basicPuppeteerExample();
};
```

- Use the Synthetics dependency.

```
var synthetics = require('Synthetics');
```

- Use the `Synthetics.getPage` function to get a Puppeteer Page object.

```
const page = await synthetics.getPage();
```

The page object returned by the `Synthetics.getPage` function has the **page.on** request, response and **requestfailed** events instrumented for logging. Synthetics also sets up HAR file generation for requests and responses on the page, and adds the canary ARN to the user-agent headers of outgoing requests on the page.

The script is now ready to be run as a Synthetics canary. Here is the updated script:

```
var synthetics = require('Synthetics'); // Synthetics dependency

const basicPuppeteerExample = async function () {
  const page = await synthetics.getPage(); // Get instrumented page from Synthetics
  await page.goto('https://example.com');
  await page.screenshot({path: '/tmp/example.png'}); // Write screenshot to /tmp folder
};

exports.handler = async () => { // Exported handler function
  return await basicPuppeteerExample();
};
```

Integrating Your Canary with Other AWS Services

All canaries can use the AWS SDK library. You can use this library when you write your canary to integrate the canary with other AWS services.

To do so, you need to add the following code to your canary. AWS For these examples, AWS Secrets Manager is used as the service that the canary is integrating with.

- Import the AWS SDK.

```
const AWS = require('aws-sdk');
```

- Create a client for the AWS service that you are integrating with.

```
const secretsManager = new AWS.SecretsManager();
```

- Use the client to make API calls to that service.

```
var params = {
  SecretId: secretName
};
return await secretsManager.getSecretValue(params).promise();
```

The following canary script code snippet demonstrates an example of integration with Secrets Manager in more detail.

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');
```

```
const AWS = require('aws-sdk');
const secretsManager = new AWS.SecretsManager();

const getSecrets = async (secretName) => {
  var params = {
    SecretId: secretName
  };
  return await secretsManager.getSecretValue(params).promise();
}

const secretsExample = async function () {
  let URL = "<URL>";
  let page = await synthetics.getPage();

  log.info(`Navigating to URL: ${URL}`);
  const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout: 30000});

  // Fetch secrets
  let secrets = await getSecrets("secretname")

  /**
   * Use secrets to login.
   *
   * Assuming secrets are stored in a JSON format like:
   * {
   *   "username": "<USERNAME>",
   *   "password": "<PASSWORD>"
   * }
   */
  let secretsObj = JSON.parse(secrets.SecretString);
  await synthetics.executeStep('login', async function () {
    await page.type(">USERNAME-INPUT-SELECTOR<", secretsObj.username);
    await page.type(">PASSWORD-INPUT-SELECTOR<", secretsObj.password);

    await Promise.all([
      page.waitForNavigation({ timeout: 30000 }),
      await page.click(">SUBMIT-BUTTON-SELECTOR<")
    ]);
  });

  // Verify login was successful
  await synthetics.executeStep('verify', async function () {
    await page.waitForXPath(">SELECTOR<", { timeout: 30000 });
  });
};

exports.handler = async () => {
  return await secretsExample();
};
```

Library Functions Available for Canary Scripts

CloudWatch Synthetics includes several built-in functions that you can call when writing Node.js scripts to use as canaries.

Some functions apply to both UI and API canaries. Others apply to UI canaries only. A UI canary is a canary that uses the `getPage()` function and uses Puppeteer as a web driver to navigate and interact with webpages.

Topics

- [Library Functions That Apply to All Canaries \(p. 116\)](#)
- [Library Functions That Apply to UI Canaries Only \(p. 117\)](#)

Library Functions That Apply to All Canaries

The following CloudWatch Synthetics library functions are useful for all canaries.

Topics

- [getLogLevel\(\);](#) (p. 116)
- [setLogLevel\(\);](#) (p. 116)
- [Synthetics Logger](#) (p. 116)

[getLogLevel\(\);](#)

Retrieves the current log level for the Synthetics library. Possible values are the following:

- 0 – Debug
- 1 – Info
- 2 – Warn
- 3 – Error

Example:

```
let logLevel = synthetics.getLogLevel();
```

[setLogLevel\(\);](#)

Sets the log level for the Synthetics library. Possible values are the following:

- 0 – Debug
- 1 – Info
- 2 – Warn
- 3 – Error

Example:

```
synthetics.setLogLevel(0);
```

Synthetics Logger

SyntheticsLogger writes logs out to both the console and to a local log file at the same log level. This log file is written to both locations only if the log level is at or below the desired logging level of the log function that was called.

The logging statements in the local log file are prepended with "DEBUG: ", "INFO: ", and so on to match the log level of the function that was called.

You can use the SyntheticsLogger assuming you want to run the Synthetics Library at the same log level as your Synthetics canary logging.

Using the SyntheticsLogger is not required to create a log file that is uploaded your S3 results location. You could instead create a different log file in the `/tmp` folder. Any files created under the `/tmp` folder are uploaded to the results location in S3 as artifacts.

To use the Synthetics Library logger:

```
const log = require('SyntheticsLogger');
```

Useful function definitions:

log.debug(*message*, *ex*);

Parameters: *message* is the message to log. *ex* is the exception, if any, to log.

Example:

```
log.debug("Starting step - login.");
```

log.error(*message*, *ex*);

Parameters: *message* is the message to log. *ex* is the exception, if any, to log.

Example:

```
try {  
  await login();  
} catch (ex) {  
  log.error("Error encountered in step - login.", ex);  
}
```

log.info(*message*, *ex*);

Parameters: *message* is the message to log. *ex* is the exception, if any, to log.

Example:

```
log.info("Successfully completed step - login.");
```

log.log(*message*, *ex*);

This is an alias for log.info.

Parameters: *message* is the message to log. *ex* is the exception, if any, to log.

Example:

```
log.log("Successfully completed step - login.");
```

log.warn(*message*, *ex*);

Parameters: *message* is the message to log. *ex* is the exception, if any, to log.

Example:

```
log.warn("Exception encountered trying to publish CloudWatch Metric.", ex);
```

Library Functions That Apply to UI Canaries Only

The following CloudWatch Synthetics library functions are useful only for UI canaries.

Topics

- [async addUserAgent\(page, userAgentString\);](#) (p. 118)
- [async executeStep\(stepName = null, functionToExecute\);](#) (p. 118)
- [getPage\(\);](#) (p. 118)
- [getRequestResponseLogHelper\(\);](#) (p. 119)
- [RequestResponseLogHelper](#) class (p. 119)
- [setRequestResponseLogHelper\(\);](#) (p. 120)

[async addUserAgent\(page, userAgentString\);](#)

This function appends *userAgentString* to the specified page's user-agent header.

Example:

```
await synthetics.addUserAgent(page, "MyApp-1.0");
```

Results in the page's user-agent header being set to *browsers-user-agent-header-valueMyApp-1.0*

[async executeStep\(stepName = null, functionToExecute\);](#)

Executes the provided step, wrapping it with start/pass/fail logging, start/pass/fail screenshots, and pass/fail and duration metrics. It also does the following:

- Logs that the step started.
- Takes a screenshot named *<stepName>-starting*.
- Starts a timer.
- Executes the provided function.
- If the function returns normally, it counts as passing. If the function throws, it counts as failing.
- End the timer.
- Logs whether the step passed or failed
- Takes a screenshot named *<stepName>-succeeded* or *<stepName>-failed*.
- Emits the *stepName SuccessPercent* metric, 100 for pass or 0 for failure.
- Emits the *stepName Duration* metric, with a value based on the step start and end times.
- Finally, returns what the *functionToExecute* returned or re-throws what *functionToExecute* threw.

Example:

```
await synthetics.executeStep('navigateToUrl', async function (timeoutInMillis = 30000) {  
    await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:  
        timeoutInMillis});});
```

Response:

Returns what *functionToExecute* returns.

[getPage\(\);](#)

Returns the current open page as a Puppeteer object. For more information, see [Puppeteer API v1.14.0](#).

Example:

```
let page = synthetics.getPage();
```

Response:

The page (Puppeteer object) that is currently open in the current browser session.

`getRequestResponseLogHelper();`

Use this function as a builder pattern for tweaking the request and response logging flags.

Example:

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper().withLogRequestHeaders(false));
```

Response:

```
{RequestResponseLogHelper}
```

RequestResponseLogHelper class

Handles the fine-grained configuration and creation of string representations of request and response payloads.

```
class RequestResponseLogHelper {  
    constructor () {  
        this.request = {url: true, resourceType: false, method: false, headers: false,  
        postData: false};  
        this.response = {status: true, statusText: true, url: true, remoteAddress: false,  
        headers: false};  
    }  
  
    withLogRequestUrl(logRequestUrl);  
  
    withLogRequestResourceType(logRequestResourceType);  
  
    withLogRequestMethod(logRequestMethod);  
  
    withLogRequestHeaders(logRequestHeaders);  
  
    withLogRequestPostData(logRequestPostData);  
  
    withLogResponseStatus(logResponseStatus);  
  
    withLogResponseStatusText(logResponseStatusText);  
  
    withLogResponseUrl(logResponseUrl);  
  
    withLogResponseRemoteAddress(logResponseRemoteAddress);  
  
    withLogResponseHeaders(logResponseHeaders);  
}
```

Example:

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper()  
    .withLogRequestPostData(true)  
    .withLogRequestHeaders(true)  
    .withLogResponseHeaders(true));
```

Response:

```
{RequestResponseLogHelper}
```

`setRequestResponseLogHelper();`

Use this function as a builder pattern for setting the request and response logging flags.

Example:

```
synthetics.setRequestResponseLogHelper().withLogRequestHeaders(true).withLogResponseHeaders(true);
```

Response:

```
{RequestResponseLogHelper}
```

Running a Canary on a VPC

You can run canaries on endpoints on a VPC and public internal endpoints. To run a canary on a VPC, you must have both the **DNS Resolution** and **DNS hostnames** options enabled on the VPC. For more information, see [Using DNS with Your VPC](#).

When you run a canary on a VPC endpoint, you must provide a way for it to send its metrics to CloudWatch and its artifacts to Amazon S3. If the VPC is already enabled for internet access, there's nothing more for you to do. The canary executes in your VPC, but can access the internet to upload its metrics and artifacts.

If the VPC is not already enabled for internet access, you have two options:

- Enable it for internet access. For more information, see [How do I give internet access to my Lambda function in a VPC?](#) on AWS Support.
- If you want to keep your VPC private, you can configure the canary to send its data to CloudWatch and Amazon S3 through private VPC endpoints. If you have not already done so, you must create endpoints for these services on your VPC. For more information, see [Using CloudWatch and CloudWatch Synthetics with Interface VPC Endpoints \(p. 447\)](#) and [Amazon VPC Endpoints for Amazon S3](#).

Troubleshooting a Canary on a VPC

If you have issues after creating or updating a canary, one of the following sections might help you troubleshoot the problem.

New Canary in Error State or Canary Can't Be Updated

If you create a canary to run on a VPC and it immediately goes into an error state, or you can't update a canary to run on a VPC, the canary's role might not have the right permissions. To run on a VPC, a canary must have the permissions `ec2:CreateNetworkInterface`, `ec2:DescribeNetworkInterface`, and `ec2>DeleteNetworkInterface`. These permissions are all contained in the `AWSLambdaVPCAccessExecutionRole` managed policy. For more information, see [Execution Role and User Permissions](#).

If this issue happened when you created a canary, you must delete the canary, and create a new one. If you use the CloudWatch console to create the new canary, under **Access Permissions**, select **Create a new role**. A new role that includes all permissions required to run the canary is created.

If this issue happens when you update a canary, you can update the canary again and provide a new role that has the required permissions.

"No test result returned" Error

If a canary displays a "no test result returned" error, one of the following issues might be the cause:

- If your VPC does not have internet access, you must use VPC endpoints to give the canary access to CloudWatch and Amazon S3. You must enable the **DNS resolution** and **DNS hostname** options in the VPC for these endpoint addresses to resolve correctly. For more information, see [Using DNS with Your VPC](#).
- Canaries must run in private subnets within a VPC. To check this, open the **Subnets** page in the VPC console. Check the subnets that you selected when configuring the canary. If they have a path to an internet gateway (**igw-**), they are not private subnets.

To help you troubleshoot these issues, see the logs for the canary.

To see the log events from a canary

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Log groups**.
3. Choose the name of the canary's log group. The log group name starts with `/aws/lambda/cwsyn-canary-name`.

Viewing Canary Statistics and Details

You can view details about your canaries and see statistics about their runs.

To be able to see all the details about your canary run results, you must be logged on to an account that has sufficient permissions. For more information, see [Required Roles and Permissions for CloudWatch Canaries](#) (p. 104).

You can also view CloudWatch metrics created by your canaries. These metrics are in the `CloudWatchSynthetics` namespace. The metrics have the names `SuccessPercent`, `Failed`, and `Duration`. These metrics have `CanaryName` as a dimension. Canaries that check a GUI workflow also have `StepName` as a dimension.

For more information about viewing CloudWatch metrics, see [Viewing Available Metrics](#) (p. 31),

To view canary statistics and details

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Canaries**.

In the details about the canaries that you have created:

- **Status** visually shows how many of your canaries have passed their most recent runs.
 - In the graph under **Canary runs**, each point represents one minute of your canaries' runs. You can pause on a point to see details.
3. To see more details about a single canary, choose a point in the **Status** graph or choose the name of the canary in the **Canaries** table.

In the details about that canary:

- Under **Canary runs**, you can choose one of the lines to see details about that run.

- Under the graph, you can choose **Screenshot**, **HAR file**, or **Logs** to see these types of details.

The logs for canary runs are stored in S3 buckets and in CloudWatch Logs.

Screenshots show how your customers view your web pages. You can use the HAR files (HTTP Archive files) to view detailed performance data about the web pages. You can analyze the list of web requests and catch performance issues such as time to load for an item. Log files show the record of interactions between the canary run and the web page and can be used to identify details of errors.

Deleting a Canary

When you delete a canary, resources used and created by the canary are not automatically deleted. After you delete a canary that you do not intend to use again, you should also delete the following:

- Lambda functions and layers used by this canary. Their prefix is `cwsyn-MyCanaryName`.
- CloudWatch alarms created for this canary. These alarms have a name of `Synthetics-Alarm-MyCanaryName`. For more information about deleting alarms, see [Editing or Deleting a CloudWatch Alarm \(p. 89\)](#).
- Amazon S3 objects and buckets, such as the canary's results location and artifact location.
- IAM roles created for the canary. These have the name `role/service-role/CloudWatchSyntheticsRole-MyCanaryName`.
- Log groups in CloudWatch Logs created for the canary. These logs groups have the following names: `/aws/lambda/cwsyn-MyCanaryName`.

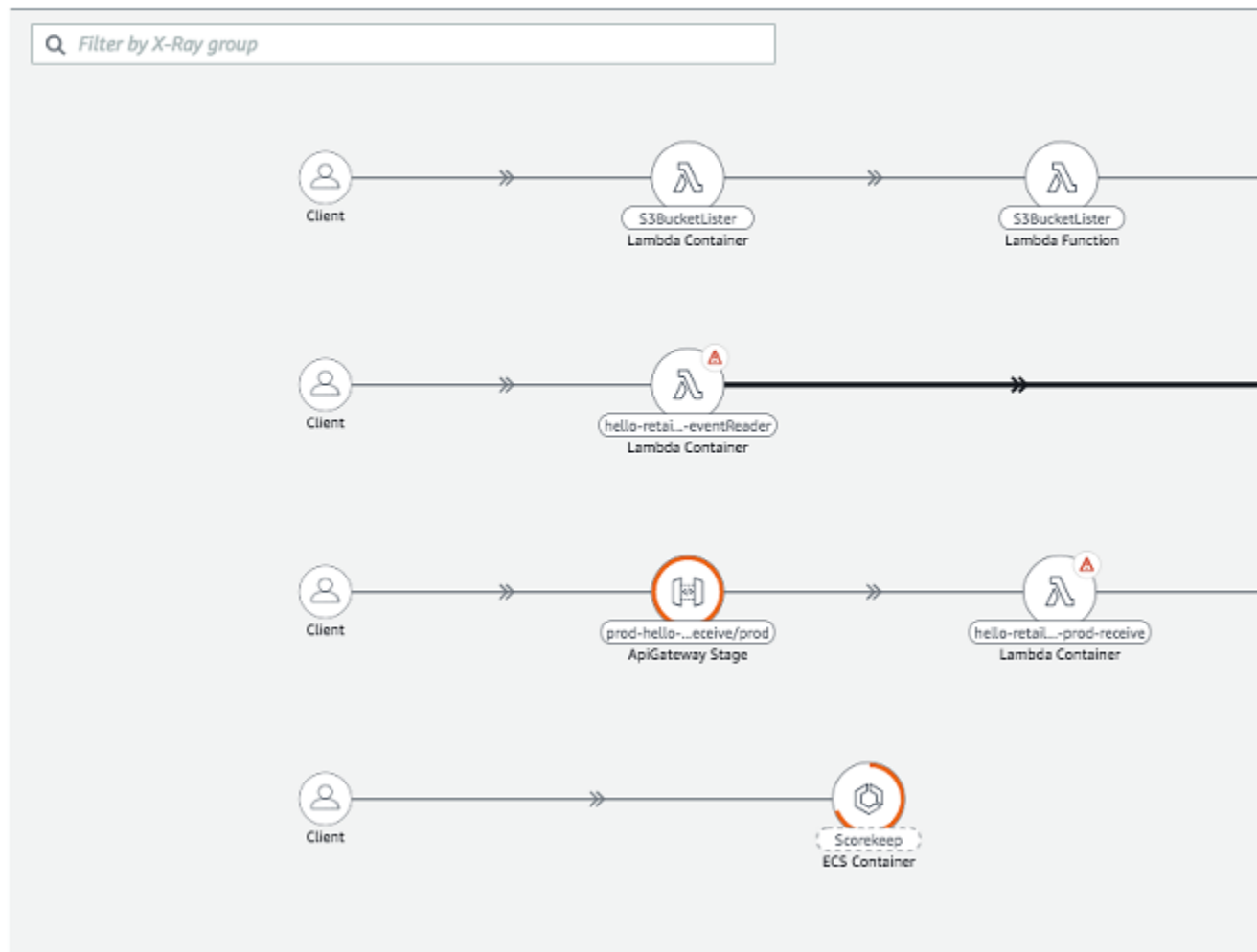
Before you delete a canary, you might want to view the canary details and make note of this information. That way, you can delete the correct resources after you delete the canary.

Using ServiceLens to Monitor the Health of Your Applications

CloudWatch ServiceLens enhances the observability of your services and applications by enabling you to integrate traces, metrics, logs, and alarms into one place. ServiceLens integrates CloudWatch with AWS X-Ray to provide an end-to-end view of your application to help you more efficiently pinpoint performance bottlenecks and identify impacted users. A service map displays your service endpoints and resources as “nodes” and highlights the traffic, latency, and errors for each node and its connections. You can choose a node to see detailed insights about the correlated metrics, logs, and traces associated with that part of the service. This enables you to investigate problems and their effect on the application.

– [CloudWatch](#) > [Service Map](#)

Service map



To fully take advantage of ServiceLens and correlated metrics, logs, and traces, you must update the X-Ray SDK and the instrumentation of your application. ServiceLens supports logs correlation for Lambda

functions, API Gateway, Java-based applications running on Amazon EC2, and Java-based applications running on Amazon EKS or Kubernetes with Container Insights deployed.

ServiceLens integrates with Amazon CloudWatch Synthetics, a fully-managed service that enables you to create canaries to monitor your endpoints and APIs from the outside-in. Canaries that you create appear on the ServiceLens service map. For more information, see [Using Synthetic Monitoring \(p. 103\)](#).

ServiceLens is available in every Region where X-Ray is available.

Topics

- [Deploying ServiceLens \(p. 124\)](#)
- [Using the Service Map in ServiceLens \(p. 132\)](#)
- [Using the Traces View in ServiceLens \(p. 134\)](#)
- [ServiceLens Troubleshooting \(p. 134\)](#)

Deploying ServiceLens

Deploying ServiceLens requires two steps:

- Deploy AWS X-Ray so that you can view the service map.
- Deploy the CloudWatch agent and the X-Ray daemon to enable the service map integration with CloudWatch metrics and CloudWatch Logs.

Topics

- [Deploying AWS X-Ray \(p. 124\)](#)
- [Deploying the CloudWatch Agent and the X-Ray Daemon \(p. 126\)](#)

Deploying AWS X-Ray

You can use any AWS X-Ray SDK to enable X-Ray. However, the correlation of logs and metrics with your traces is supported only if you use the Java SDK.

To deploy X-Ray, follow the standard X-Ray setup. For more information, see the following:

- [AWS X-Ray SDK for Java](#) (supports logs correlations)
- [The X-Ray SDK for Node.js](#)
- [AWS X-Ray SDK for .NET](#)
- [AWS X-Ray SDK for Go](#)
- [AWS X-Ray SDK for Python](#)
- [AWS X-Ray SDK for Ruby](#)

After completing the X-Ray setup, follow the steps in the following sections to integrate X-Ray with CloudWatch Logs and enable segment metrics.

Topics

- [Integrating With CloudWatch Logs \(p. 125\)](#)
- [Enabling Segment Metrics From X-Ray \(p. 125\)](#)

Integrating With CloudWatch Logs

To enable integration with CloudWatch Logs, there are two steps:

- Enable trace to logs correlation. This is supported only using the SDK for Java.
- Configure trace ID injection.

Enabling Trace to Logs Correlation

The SDK for Java supports both a set of standard application logging frameworks and CloudWatch Logs native support. Before completing the following steps, you must have completed a standard setup of the AWS X-Ray SDK for Java.

The supported runtimes are Amazon EC2, Amazon ECS with CloudWatch Container Insights enabled, and Lambda.

- To enable trace to logs correlation on Amazon EC2, enable the X-Ray EC2 Plugin. For more information, see [Service Plugins](#)
- To enable trace to logs correlation on Amazon EKS, first enable Container Insights if you have not already done so. For more information, see [Using Container Insights \(p. 155\)](#).

Then, enable the X-Ray SDK EKS Plugin. For more information, see [Service Plugins](#).

- To enable trace to logs correlation on Lambda, you must enable X-Ray on Lambda. For more information, see [AWS Lambda and AWS X-Ray](#).

Enabling Trace ID Injection

For information about how to enable trace ID injection, see [Logging](#).

Enabling Segment Metrics From X-Ray

The AWS X-Ray SDK for Java can emit several metrics about segments into CloudWatch to give an unsampled view of latency, throttle, error, and fault rates. It uses the CloudWatch agent to emit these metrics to minimize the impact on application performance. For more information about segments, see [Segments](#).

If you enable segment metrics, a log group called **XRayApplicationMetrics** is created, and the metrics **ErrorRate**, **FaultRate**, **ThrottleRate**, and **Latency**, are published into a custom CloudWatch metric namespace called **Observability**.

Segment metrics are not currently supported in Lambda.

To enable the AWS X-Ray SDK for Java to publish segment metrics, use the following example.

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withSegmentListener(new MetricsSegmentListener());
```

If you are using ServiceLens with Amazon EKS and Container Insights, add the `AWS_XRAY_METRICS_DAEMON_ADDRESS` environment variable to the `HOST_IP` as shown in the following example.

```
env:
- name: HOST_IP
  valueFrom:
    fieldRef:
      apiVersion: v1
```

```
    fieldPath: status.hostIP
- name: AWS_XRAY_METRICS_DAEMON_ADDRESS
  value: $(HOST_IP):25888
```

For more information, see [Enable X-Ray CloudWatch Metrics](#).

Deploying the CloudWatch Agent and the X-Ray Daemon

This section explains how to deploy the CloudWatch agent and the X-Ray daemon. You can deploy the agent and the daemon in the following environments:

- Amazon ECS or Fargate
- Amazon EKS or Kubernetes hosted on Amazon EC2
- Amazon EC2

The deployment steps for each of these environments are explained in the following sections.

Topics

- [Deploying the CloudWatch Agent and the X-Ray Daemon on Amazon ECS \(p. 126\)](#)
- [Deploying the CloudWatch Agent and the X-Ray Daemon on Amazon EKS or Kubernetes \(p. 130\)](#)
- [Deploying the CloudWatch Agent and the X-Ray Daemon on Amazon EC2 \(p. 132\)](#)

Deploying the CloudWatch Agent and the X-Ray Daemon on Amazon ECS

On Amazon ECS, you deploy the CloudWatch agent as a sidecar to your application container to collect metrics. You can configure the CloudWatch Agent through SSM parameter store.

Creating IAM Roles

You must create two IAM roles. If you already have created these roles, you may need to add permissions to them.

- **ECS task role**— Containers use this role to run. The permissions should be whatever your applications need, plus **CloudWatchAgentServerPolicy** and **AWSXRayDaemonWriteAccess**.
- **ECS task definition role**— Amazon ECS uses this role to launch and execute your containers. If you have already created this role, attach the **AmazonSSMReadOnlyAccess**, **AmazonECSTaskExecutionRolePolicy**, and **CloudWatchAgentServerPolicy** policies to it.

If you need to store more sensitive data for Amazon ECS to use, see [Specifying Sensitive Data](#) for more information.

For more information about creating IAM roles, see [Creating IAM Roles](#).

Store the Agent Configuration in SSM Parameter Store

You need to make sure your agent configuration file has the following section, and then upload it to the SSM parameter store.

```
{
  "logs": {
```

```
"metrics_collected": {  
  "emf": {}  
}
```

To upload the agent configuration to the SSM parameter store

1. Put the agent configuration content into a local file `/tmp/ecs-cwagent.json`.
2. Enter the following command. Replace `region` with the Region of your cluster.

```
aws ssm put-parameter \  
--name "ecs-cwagent" \  
--type "String" \  
--value "`cat /tmp/ecs-cwagent.json`" \  
--region "region"
```

Create a Task Definition and Launch the Task

The steps for this task depend on whether you want to use the EC2 launch type or the Fargate launch type.

EC2 Launch Type

First, create the task definition. In this example, the container “demo-app” sends X-Ray SDK metrics to the CloudWatch agent and sends trace information to the X-Ray daemon.

Copy the following task definition to a local JSON file such as `/tmp/ecs-cwagent-ec2.json`. Replace the following placeholders:

- Replace `{{ecs-task-role}}` with the ARN of your ECS task role.
- Replace `{{ecs-task-execution-role}}` with the ARN of your ECS task execution role.
- Replace `{{demo-app-image}}` with your application image that has X-Ray SDK integration enabled. Change the name from demo-app to your own application name.
- Replace `{{region}}` with the name of the AWS Region where you want to send the logs for containers. For example, `us-west-2`.

```
{  
  "family": "ecs-cwagent-ec2",  
  "taskRoleArn": "{{ecs-task-role}}",  
  "executionRoleArn": "{{ecs-task-execution-role}}",  
  "networkMode": "bridge",  
  "containerDefinitions": [  
    {  
      "name": "demo-app",  
      "image": "{{demo-app-image}}",  
      "links": [  
        "cloudwatch-agent",  
        "xray-daemon"  
      ],  
      "logConfiguration": {  
        "logDriver": "awslogs",  
        "options": {  
          "awslogs-create-group": "True",  
          "awslogs-group": "/ecs/ecs-cwagent-ec2",  
          "awslogs-region": "{{region}}",  
          "awslogs-stream-prefix": "ecs"  
        }  
      }  
    }  
  ]  
}
```

```
    },
    {
      "name": "xray-daemon",
      "image": "amazon/aws-xray-daemon:latest",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-ec2",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    },
    {
      "name": "cloudwatch-agent",
      "image": "amazon/cloudwatch-agent:latest",
      "secrets": [
        {
          "name": "CW_CONFIG_CONTENT",
          "valueFrom": "ecs-cwagent"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-ec2",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ],
  "requiresCompatibilities": [
    "EC2"
  ],
  "cpu": "256",
  "memory": "256"
}
```

Enter the following command to create the task definition. Replace `{{region}}` with the Region of your cluster.

```
aws ecs register-task-definition \
  --cli-input-json file:///tmp/ecs-cwagent-ec2.json \
  --region {{region}}
```

Enter the following command to launch the task. Replace `{{cluster-name}}` and `{{region}}` with the name and Region of your cluster.

```
aws ecs run-task \
  --cluster {{cluster-name}} \
  --task-definition ecs-cwagent-ec2 \
  --region {{region}} \
  --launch-type EC2
```

Fargate Launch Type

First, create the task definition. In this example, the container “demo-app” sends X-Ray SDK metrics to the CloudWatch agent and sends trace information to the X-Ray daemon.

Copy the following task definition to a local JSON file such as `/tmp/ecs-cwagent-ec2.json`. Replace the following placeholders:

- Replace `{{ecs-task-role}}` with the Amazon Resource Name (ARN) of your ECS task role.
- Replace `{{ecs-task-execution-role}}` with the ARN of your ECS task execution role.
- Replace `{{demo-app-image}}` with your application image that has X-Ray SDK integration enabled. Change the name from `demo-app` to your own application name.
- Replace `{{region}}` with the name of the AWS Region where you want to send the logs for containers. For example, `us-west-2`.

```
{
  "family": "ecs-cwagent-fargate",
  "taskRoleArn": "{{ecs-task-role}}",
  "executionRoleArn": "{{ecs-task-execution-role}}",
  "networkMode": "awsvpc",
  "containerDefinitions": [
    {
      "name": "demo-app",
      "image": "{{demo-app-image}}",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-fargate",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    },
    {
      "name": "xray-daemon",
      "image": "amazon/aws-xray-daemon:latest",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-fargate",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    },
    {
      "name": "cloudwatch-agent",
      "image": "amazon/cloudwatch-agent:latest",
      "secrets": [
        {
          "name": "CW_CONFIG_CONTENT",
          "valueFrom": "ecs-cwagent"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-create-group": "True",
          "awslogs-group": "/ecs/ecs-cwagent-fargate",
          "awslogs-region": "{{region}}",
          "awslogs-stream-prefix": "ecs"
        }
      }
    }
  ]
}
```

```
"requiresCompatibilities": [  
    "FARGATE"  
],  
"cpu": "512",  
"memory": "1024"  
}
```

Enter the following command to create the task definition. Replace `{{region}}` with the Region of your cluster.

```
aws ecs register-task-definition \  
  --cli-input-json file:///tmp/ecs-cwagent-fargate.json \  
  --region {{region}}
```

If you already have a Fargate cluster set up, you can use the task definition you just created to launch the task. If you do not yet have any Fargate clusters, see [Configure the Service](#) for more information about the rest of the steps to set up Fargate.

Deploying the CloudWatch Agent and the X-Ray Daemon on Amazon EKS or Kubernetes

These topics explain how to install the X-Ray daemon and the CloudWatch agent on Amazon EKS or Kubernetes.

Deploying the X-Ray Daemon on Amazon EKS or Kubernetes

To install the CloudWatch agent and the X-Ray daemon on Amazon EKS or Kubernetes, you can use a quick setup.

To install the CloudWatch agent and the X-Ray daemon on Amazon EKS or Kubernetes

1. Ensure that the IAM role that is attached to the EC2 instance, or the Kubernetes worker node, has the **CloudWatchAgentServerPolicy** and **AWSXRayDaemonWriteAccess** policies attached.
2. Enter the following command:

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/master/k8s-deployment-manifest-templates/deployment-mode/daemonset/cwagent-fluentd-xray/cwagent-fluentd-xray-quickstart.yaml | sed "s/{{cluster_name}}/cluster-name/;s/{{region_name}}/region/" | kubectl apply -f -
```

What the Quick Start Does

This section describes the quick setup of the CloudWatch agent and the X-Ray daemon.

- The quick setup installs the X-Ray daemon via the `kubectl apply -f` command with the following file content.

```
apiVersion: apps/v1  
kind: DaemonSet  
metadata:  
  name: xray-daemon  
  namespace: amazon-cloudwatch  
spec:  
  selector:  
    matchLabels:  
      name: xray-daemon
```

```
template:
  metadata:
    labels:
      name: xray-daemon
  spec:
    containers:
      - name: xray-daemon
        image: amazon/aws-xray-daemon:latest
        imagePullPolicy: Always
        ports:
          - containerPort: 2000
            hostPort: 2000
            protocol: UDP
        resources:
          limits:
            cpu: 100m
            memory: 256Mi
          requests:
            cpu: 50m
            memory: 50Mi
    terminationGracePeriodSeconds: 60
```

- The quick setup updates the CloudWatch agent using the Docker image version/label 1.231221.0 or later, or the latest version. You can find the image at <https://hub.docker.com/r/amazon/cloudwatch-agent>.
- To enable the X-Ray SDK to read cluster name and Region information, the quick setup updates the CloudWatch agent using the Docker image version/label 1.231221.0, or later, or the latest. You can find the image at <https://hub.docker.com/r/amazon/cloudwatch-agent>.
- To enable the X-Ray SDK to read cluster name and Region information, the quick setup created a file with the following content, and then applied it with the **kubectl apply -f** command.

```
---
# create role binding for XRay SDK to read config map
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: container-insights-discovery-role
  namespace: amazon-cloudwatch
rules:
  - apiGroups:
    - ""
  resourceNames:
    - cluster-info
  resources:
    - configmaps
  verbs:
    - get
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: service-users-cloudwatch-discovery-role-binding
  namespace: amazon-cloudwatch
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: container-insights-discovery-role
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: Group
    name: system:serviceaccounts
```

- The quick setup exposes the CloudWatch agent port that receives X-Ray SDK metrics. The default port is UDP 25888.

```
ports:
  - containerPort: 25888
    hostPort: 25888
    protocol: UDP
```

- The quick setup merges the agent configuration JSON with the X-Ray SDK metrics configuration with the following JSON.

```
{
  "logs": {
    "metrics_collected": {
      "emf": {}
    }
  }
}
```

Deploying the CloudWatch Agent and the X-Ray Daemon on Amazon EC2

Standard installations of the CloudWatch agent and the X-Ray daemon are sufficient to enable ServiceLens on Amazon EC2, with the addition of the following CloudWatch agent configuration section example. For more information about installing the agent, see [Installing the CloudWatch Agent \(p. 237\)](#). For more information about the CloudWatch agent configuration file, see [Manually Create or Edit the CloudWatch Agent Configuration File \(p. 275\)](#).

When you configure the CloudWatch agent, include this section in your configuration file:

```
{
  "logs": {
    "metrics_collected": {
      "emf": {}
    }
  }
}
```

For more information about installing the X-Ray daemon, see [X-Ray Daemon Configuration](#).

Using the Service Map in ServiceLens

This section introduces the service map and helps you learn to navigate it.

To see a service map, you must have installed AWS X-Ray and completed the other ServiceLens deployment steps. For more information, see [Deploying ServiceLens \(p. 124\)](#).

You must also be logged on to an account that has the `AWSXrayReadOnlyAccess` managed policy, as well as permissions that enable you to view the CloudWatch console. For more information, see [How AWS X-Ray Works with IAM](#) and [Using Amazon CloudWatch Dashboards \(p. 16\)](#).

To begin using the service map

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

2. In the navigation pane, choose **Service Map**.

A service map appears. It has the following parts:

- The AWS services and your custom applications that you have enabled tracing for are shown as circles or "nodes." The size of each node indicates the relative number of traced requests that are going to that service.
- Edges, or connections between nodes, are shown as lines connecting the nodes. By default, the thickness of a line indicates the relative number of traced requests between those nodes.

You can use the dropdown menu in the top right to choose whether the number of traced requests or the average latency is used for node and edge sizing. You can also select to use constant size for all nodes and edges.

- The entry point to your nodes is shown on the left as a "Client." A "Client" represents both web server traffic and traced API operation requests.
 - A node outlined partially in red, orange, or purple has issues. Some traced requests to these nodes have faults, errors, or throttling. The percentage of the color outline indicates the percentage of traced requests that are having issues.
 - If a node has a triangle with an exclamation point next to it, at least one CloudWatch alarm related to that node is in alarm state.
3. By default, the data in the map is for the most recent 6-hour time window. To change the timeframe of the window, use the controls at the upper right of the screen. The time range to be shown can be up to 6 hours, and can be as much as 30 days in the past.
 4. If you have enabled X-Ray groups, you can filter the map by selecting an X-ray group in the filter.
 5. To focus on the incoming and outgoing connections for a node, select the node and choose **View connections** near the top of the service map.
 6. To see a pop-up displaying latency, errors, requests, and alarm summary statistics for a node, pause on that node.
 7. To see latency statistics for an edge connection, pause on the line representing that edge.
 8. To display alarm status for a service, along with line charts for latency, errors, and trace counts, choose that service node on the map.

For more information about this view, see the following procedures.

9. To view the service map as a table, choose **List view** near the top of the screen. In this view, you can filter and sort the nodes and alarms that are displayed on the map.
10. To see a dashboard with metrics for a specific node, select the node and then choose **View dashboard** near the bottom of the screen.

To view traces for a service or application on the service map

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Service Map**.
3. Choose the node that represents the service or application that you want to investigate.

CloudWatch displays line charts of latency, errors, and trace counts for that service, along with a summary of alarm status.

Above those charts are options to dive down to logs and traces for the service.

4. To view traces related to the service, choose **View traces**.

The console switches to the **Traces** view, focused on the service that you are investigating. For more information, see [Using the Traces View in ServiceLens \(p. 134\)](#).

Using the Traces View in ServiceLens

The traces view enables you to view recent X-Ray traces in your application.

To view traces and dive down for more information

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Traces**.
3. Under **Filter type**, you can choose different criteria to sort the traces by. As you do, CloudWatch displays the success rate and response time of the traces according to your choice. The list of traces displayed at the bottom of the page is narrowed to traces that match your filter. The first 1000 traces that match your filter are retrieved.

The values in the filter table are populated by the traces that the filter returns, and update as you refine your choices in the filter.

Use the **Custom query** option under **Filter type** to add a custom expression as a filter. Custom expressions can contain keywords, unary or binary operators, and comparison values for the keywords. Keywords can correspond to parts of a trace, such as `responsetime`, `duration`, and status codes. For more information, see [Filter Expression Syntax](#).

You can focus your filter further by selecting the check box next to a row under **Traces by** and choosing **Add to filter**. The filter value from the selected row is added to the filter.

4. To see a histogram representing the response time distribution of the traces returned by the current filter, and a table displaying each individual trace, scroll to the bottom of the screen.

Choose one of the traces in the table to view detailed information about the trace.

The trace details page displays a timeline that includes each of the segments that comprise the trace. Choose a segment to view additional associated metadata, including errors where applicable. For traces where the logs can be associated, the log lines associated with the trace are displayed under the timeline.

You can then optionally view those log entries in CloudWatch Logs Insights. To do that, choose **View in CloudWatch Logs Insights** and then choose **Run query**.

Some traces do not have associated log entries.

- If you don't see log entries for Amazon EC2 or Amazon EKS, you need to update to the latest version of the X-Ray SDK. For more information, see [Deploying AWS X-Ray \(p. 124\)](#).
- ServiceLens shows log entries for Amazon ECS only if there are 20 or fewer log groups with names that start with `/ecs/`.
- Currently, log entries for Fargate traces are not available.

ServiceLens Troubleshooting

The following sections can help if you're having issues with CloudWatch ServiceLens.

I Don't See All My Logs

How to configure logs to appear in ServiceLens depends on the service.

- API Gateway logs appear if logging is turned on in API Gateway.
- Amazon ECS and Amazon EKS logs appear if you are using the latest versions of the X-Ray SDK and the CloudWatch agent. For more information, see [Deploying ServiceLens \(p. 124\)](#).

- Lambda logs appear if the request ID is in the log entry. This happens automatically for the situations listed in the following table. For other cases, where the runtime does not automatically include the trace ID, you can manually include the trace ID.

Runtime	Method	Request ID Automatically in Log Entry?
Java	context.getLogger.log aws-lambda-java-log4j2	Yes
Java	System.out.println	No
Python	context.log logging.info/error/log/etc...	Yes
Python	print	No
Node.js	context.log console.log/info/error/etc...	Yes
dotnet	context.Logger.log Console.WriteLine()	No
Go	fmt.Printf log.Print	No
Ruby	puts	No

I Don't See All My Alarms on the Service Map

ServiceLens shows only the alert icon for a node if any alarms associated with that node are in the ALARM state.

ServiceLens associates alarms with nodes using the following logic:

- If the node represents an AWS service, then all alarms with the namespace associated with that service are associated with the node. For example, a node of type `AWS::Kinesis` is linked with all alarms that are based on metrics in the CloudWatch namespace `AWS/Kinesis`.
- If the node represents an AWS resource, then the alarms on that specific resource are linked. For example, a node of type `AWS::DynamoDB::Table` with the name "MyTable" is linked to all alarms that are based on a metric with the namespace `AWS/DynamoDB` and have the `TableName` dimension set to `MyTable`.
- If the node is of unknown type, which is identified by a dashed border around the name, then no alarms are associated with that node.

I Don't See Some AWS Resources on the Service Map

For AWS resources to be traced on the service map, the AWS SDK must be captured using the X-Ray SDK. For more information about X-Ray, see [What Is AWS X-Ray](#).

Not every AWS resource is represented by a dedicated node. Some AWS services are represented by a single node for all requests to the service. The following resource types are displayed with a node per resource:

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

Lambda functions are represented by two nodes— one for the Lambda Container, and one for the function. This helps to identify cold start problems with Lambda functions. Lambda container nodes are associated with alarms and dashboards in the same way as Lambda function nodes.

- `AWS::ApiGateway::Stage`
- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

There Are Too Many Nodes on My Service Map

Use X-Ray groups to break your map into multiple maps. For more information, see [Using Filter Expressions with Groups](#).

Cross-Account Cross-Region CloudWatch Console

You can add *cross-account* functionality to your CloudWatch console. This functionality provides you with cross-account visibility to your dashboards, alarms, metrics, and automatic dashboards without having to log in and log out of different accounts.

You can then create dashboards that summarize CloudWatch data from multiple AWS accounts and multiple AWS Regions into a single dashboard.

Many organizations have their AWS resources deployed in multiple accounts, to provide billing and security boundaries. In this case, we recommend that you designate one or more of your accounts as your monitoring accounts, and build your cross-account dashboards in these accounts.

Cross-account functionality is integrated with AWS Organizations, to help you efficiently build your cross-account dashboards.

Cross-Region Functionality

Cross-Region functionality is now built-in automatically. You do not need to take any extra steps to be able to display metrics from different Regions in a single account on the same graph or the same dashboard.

Enabling Cross-Account Functionality in CloudWatch

To set up cross-account functionality in your CloudWatch console, use the AWS Management Console to set up your sharing accounts and monitoring accounts.

Set Up A Sharing Account

You must enable sharing in each account that will make data available to the monitoring account.

To enable your account to share CloudWatch data with other accounts

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Settings**, then choose **Configure**.
3. Choose **Share data**.
4. For **Sharing**, choose **Specific accounts** and enter the IDs of the accounts that you want to share data with.

Any accounts that you specify here can view your account's CloudWatch data. Specify the IDs only of accounts that you know and trust.

5. For **Permissions**, specify how to share your data with one of the following options:
 - **Provide read-only access to your CloudWatch metrics, dashboards, and alarms.** This option enables the monitoring accounts to create cross-account dashboards that include widgets that contain CloudWatch data from your account.
 - **Include CloudWatch automatic dashboards.** If you select this option, users in the monitoring account can also view the information in this account's automatic dashboards. For more information, see [Getting Started with Amazon CloudWatch \(p. 10\)](#).

- **Include X-Ray read-only access for ServiceLens.** If you select this option, users in the monitoring account can also view the ServiceLens service map and X-Ray trace information in this account. For more information, see [Using ServiceLens to Monitor the Health of Your Applications \(p. 123\)](#).
 - **Full read-only access to everything in your account.** This option enables the accounts that you use for sharing to create cross-account dashboards that include widgets that contain CloudWatch data from your account. It also enables those accounts to look deeper into your account and view your account's data in the consoles of other AWS services.
6. Choose **Launch CloudFormation template**.

In the confirmation screen, type **Confirm**, and choose **Launch template**.

7. Select the **I acknowledge...** check box, and choose **Create stack**.

Sharing With an Entire Organization

Completing the preceding procedure creates an IAM role which enables your account to share data with one account. You can create or edit an IAM role that shares your data with all accounts in an organization. Do this only if you know and trust all accounts in the organization.

To share your CloudWatch account data with all accounts in an organization

1. If you haven't already, complete the preceding procedure to share your data with one AWS account.
2. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
3. In the navigation pane, choose **Roles**.
4. In the list of roles, choose **CloudWatch-CrossAccountSharingRole**.
5. Choose **Trust relationships, Edit trust relationship**.

You see a policy like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. Change the policy to the following, replacing **org-id** with the ID of your organization.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "org-id"
        }
      }
    }
  ]
}
```

```
}  
  }  
}
```

7. Choose **Update Trust Policy**.

Set Up a Monitoring Account

Enable each monitoring account if you want to view cross-account CloudWatch data.

When you complete the following procedure, CloudWatch creates a service-linked role that CloudWatch uses in the monitoring account to access data shared from your other accounts. This service-linked role is called **AWSServiceRoleForCloudWatchCrossAccount**. For more information, see [Using Service-Linked Roles for CloudWatch \(p. 432\)](#).

To enable your account to view cross-account CloudWatch data

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Settings**, then choose **Configure**.
3. Under **View cross-account cross-region**, choose one of the following options:
 - **Account Id Input**. This option prompts you to manually input an account ID each time that you want to switch accounts when you view cross-account data.
 - **AWS Organization account selector**. This option causes the accounts that you specified when you completed your cross-account integration with Organizations to appear. When you next use the console, CloudWatch displays a dropdown list of these accounts for you to select from when you are viewing cross-account data.

To do this, you must have first used your organization master account to allow CloudWatch to see a list of accounts in your organization. For more information, see [\(Optional\) Integrate With AWS Organizations \(p. 139\)](#).

- **Custom account selector**. This option prompts you to enter a list of account IDs. When you next use the console, CloudWatch displays a dropdown list of these accounts for you to select from when you are viewing cross-account data.

You can also enter a label for each of these accounts to help you identify them when choosing accounts to view.

The account selector settings that a user makes here are retained only for that user, not for all other users in the monitoring account.

4. Choose **Enable**.

After you complete this setup, you can create cross-account dashboards. For more information, see [Cross-Account Cross-Region Dashboards \(p. 17\)](#).

(Optional) Integrate With AWS Organizations

If you want to integrate cross-account functionality with AWS Organizations, you must make a list of all accounts in the organization available to the monitoring accounts.

To enable cross-account CloudWatch functionality to access a list of all accounts in your organization

1. Log in to your organization's master account.

2. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
3. In the navigation pane, choose **Settings**, then choose **Configure**.
4. For **Grant permission to view the list of accounts in the organization**, choose **Specific accounts** to be prompted to enter a list of account IDs. The list of accounts in your organization are shared with only the accounts that you specify here.
5. Choose **Share organization account list**.
6. Choose **Launch CloudFormation template**.

In the confirmation screen, type **Confirm**, and choose **Launch template**.

Troubleshooting Your CloudWatch Cross-Account Setup

This section contains troubleshooting tips for cross-account, console deployment in CloudWatch.

I am getting access denied errors displaying cross-account data

Check the following:

- Your monitoring account should have a role named **AWSServiceRoleForCloudWatchCrossAccount**. If it does not, you need to create this role. For more information, see [Set Up a Monitoring Account \(p. 139\)](#).
- Each sharing account should have a role named **CloudWatch-CrossAccountSharingRole**. If it does not, you need to create this role. For more information, see [Set Up A Sharing Account \(p. 137\)](#).
- The sharing role must trust the monitoring account.

To confirm that your roles are set up properly for the CloudWatch cross-account console

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. In the list of roles, make sure the needed role exists. In a sharing account, look for **CloudWatch-CrossAccountSharingRole**. In a monitoring account, look for **AWSServiceRoleForCloudWatchCrossAccount**.
4. If you are in a sharing account and **CloudWatch-CrossAccountSharingRole** already exists, choose **CloudWatch-CrossAccountSharingRole**.
5. Choose **Trust relationships**, **Edit trust relationship**.
6. Confirm that the policy lists either the account ID of the monitoring account, or the organization ID of an organization that contains the monitoring account.

I don't see an account drop-down in the console

First, check that you have created the correct IAM roles, as discussed in the preceding troubleshooting section. If those are set up correctly, make sure that you have enabled this account to view cross-account data, as described in [Enable Your Account to View Cross-Account Data \(p. 139\)](#).

Using CloudWatch Anomaly Detection

When you enable *anomaly detection* for a metric, CloudWatch applies statistical and machine learning algorithms. These algorithms continuously analyze metrics of systems and applications, determine normal baselines, and surface anomalies with minimal user intervention.

The algorithms generate an anomaly detection model. The model generates a range of expected values that represent normal metric behavior.

You can use the model of expected values in two ways:

- Create anomaly detection alarms based on a metric's expected value. These types of alarms don't have a static threshold for determining alarm state. Instead, they compare the metric's value to the expected value based on the anomaly detection model.

You can choose whether the alarm is triggered when the metric value is above the band of expected values, below the band, or both.

For more information, see [Creating a CloudWatch Alarm Based on Anomaly Detection \(p. 82\)](#).

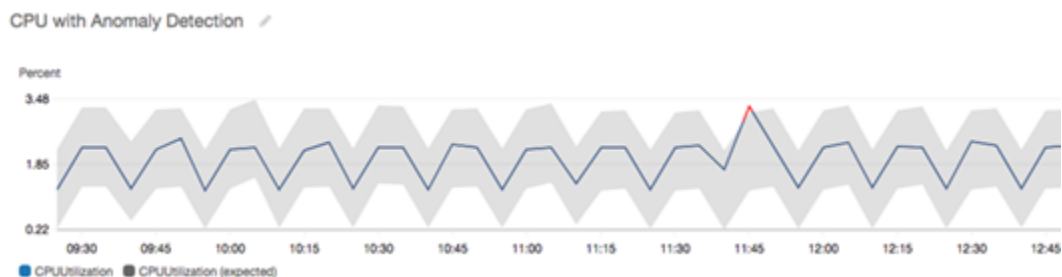
- When viewing a graph of metric data, overlay the expected values onto the graph as a band. This makes it visually clear which values in the graph are out of the normal range. For more information, see [Creating a Graph \(p. 43\)](#).

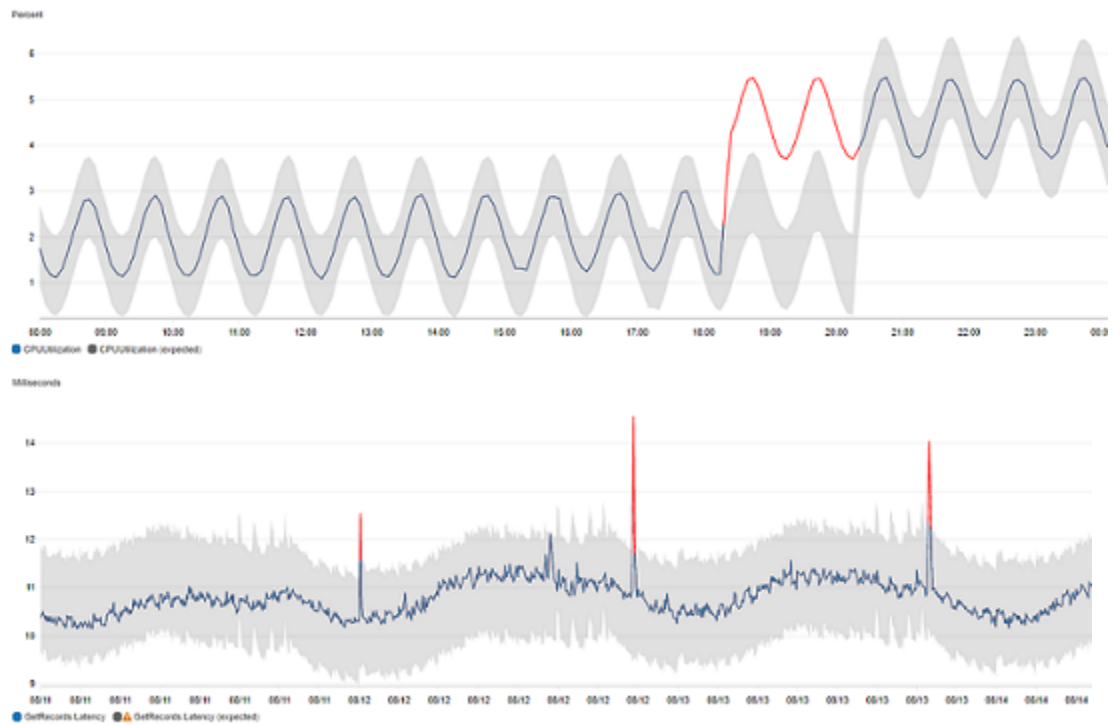
You can enable anomaly detection using the AWS Management Console, the AWS CLI, AWS CloudFormation, or the AWS SDK. You can enable anomaly detection on metrics vended by AWS and also on custom metrics.

You can also retrieve the upper and lower values of the model's band by using the `GetMetricData` API request with the `ANOMALY_DETECTION_BAND` metric math function. For more information, see [GetMetricData](#).

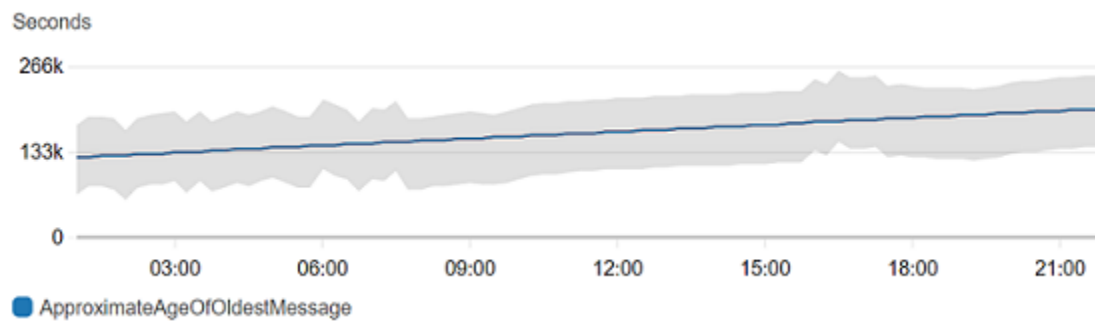
In a graph with anomaly detection, the expected range of values is shown as a gray band. If the metric's actual value goes beyond this band, it is shown as red during that time.

Anomaly detection algorithms account for the seasonality and trend changes of metrics. The seasonality changes could be hourly, daily, or weekly, as shown in the following examples.

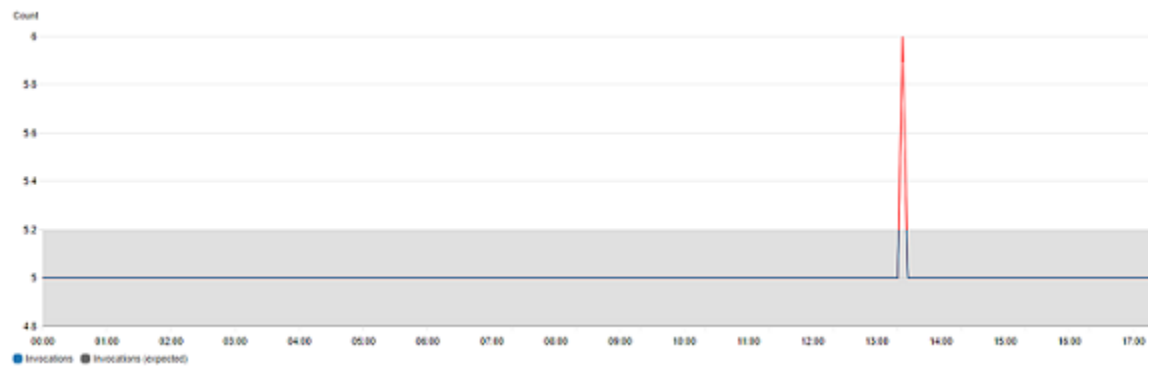




The longer-range trends could be downward or upward.



Anomaly detections also works well with metrics with flat patterns.



How CloudWatch Anomaly Detection Works

When you enable anomaly detection for a metric, CloudWatch applies machine learning algorithms to the metric's past data to create a model of the metric's expected values. The model assesses both trends and hourly, daily, and weekly patterns of the metric. The algorithm trains on up to two weeks of metric data, but you can enable anomaly detection on a metric even if the metric does not have a full two weeks of data.

You specify a value for the anomaly detection threshold that CloudWatch uses along with the model to determine the "normal" range of values for the metric. A higher value for the anomaly detection threshold produces a thicker band of "normal" values.

The machine learning model is specific to a metric and a statistic. For example, if you enable anomaly detection for a metric using the `AVG` statistic, the model is specific to the `AVG` statistic.

After you create a model, it constantly updates itself, using the latest data from the metric.

After you enable anomaly detection on a metric, you can choose to exclude specified time periods of the metric from being used to train the model. This way, you can exclude deployments or other unusual events from being used for model training, ensuring the most accurate model is created.

Using anomaly detection models for alarms incurs charges on your AWS account. For more information, see [Amazon CloudWatch Pricing](#).

Using Contributor Insights to Analyze High-Cardinality Data

You can use Contributor Insights to analyze log data and create time series that display contributor data. You can see metrics about the top-N contributors, the total number of unique contributors, and their usage. This helps you find top talkers and understand who or what is impacting system performance. For example, you can find bad hosts, identify the heaviest network users, or find the URLs that generate the most errors.

You can build your rules from scratch, and when you use the AWS Management Console you can also use sample rules that AWS has created. Rules define the log fields that you want to use to define contributors, such as `IpAddress`. You can also filter the log data to find and analyze the behavior of individual contributors.

CloudWatch also provides built-in rules that you can use to analyze metrics from other AWS services. Currently, built-in rules are available for Amazon DynamoDB.

All rules analyze incoming data in real time.

Note

If you use Contributor Insights, you are charged for each occurrence of a log event that matches a rule. For more information, see [Amazon CloudWatch Pricing](#).

Topics

- [Creating a Contributor Insights Rule](#) (p. 144)
- [Contributor Insights Rule Syntax](#) (p. 147)
- [Contributor Insights Rule Examples](#) (p. 149)
- [Viewing Contributor Insights Reports](#) (p. 151)
- [Graphing Metrics Generated by Rules](#) (p. 152)
- [Using Contributor Insights Built-In Rules](#) (p. 154)

Creating a Contributor Insights Rule

You can create rules to analyze log data. Any logs in JSON or Common Log Format (CLF) can be evaluated. This includes your custom logs that follow one of these formats and logs from AWS services such as Amazon VPC flow logs, Amazon Route 53 DNS query logs, Amazon ECS container logs, and logs from AWS CloudTrail, Amazon SageMaker, Amazon RDS, AWS AppSync and API Gateway.

In a rule, when you specify field names or values, all matching is case sensitive.

You can use built-in sample rules when you create a rule or you can create your own rule from scratch.

To create a rule using a built-in sample rule

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Contributor Insights**.
3. Choose **Create rule**.

4. Choose **Sample rule**, and from **Select sample rule**, select the rule.
5. For **Rule name**, enter a name. Valid characters are A-Z, a-z, 0-9, "-", "_", and ".".
6. For **Log group(s)**, select the log groups that you want the rule to monitor. You can select as many as 20 log groups.

To select all log groups with names that start with a certain string, choose **Select by prefix match** and enter the prefix.

If you use **Select by prefix match**, be aware of how many log groups will match your prefix and be analyzed by the rule. You incur charges for each log event that matches a rule. If you accidentally search more log groups than you intend, you might incur unexpected charges. For more information, see [Amazon CloudWatch Pricing](#).

7. The sample rule has filled out the **Fields**, **Contribution**, **Filters**, and **Aggregate on** fields. You can adjust those values, if you like.
8. Choose whether to create the rule in a disabled or enabled state. If you choose to enable it, the rule immediately starts analyzing your data. You incur costs when you run enabled rules. For more information, see [Amazon CloudWatch Pricing](#).

Contributor Insights analyzes only new log events after a rule is created. A rule cannot process logs events that were previously processed by CloudWatch Logs.

9. Choose **Create**.

To create a rule from scratch

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Contributor Insights**.
3. Choose **Create rule**.
4. In the **Create rule** wizard, choose **Custom rule**.
5. For **Rule name**, enter a name. Valid characters are A-Z, a-z, 0-9, "-", "_", and ".".
6. You can finish creating the rule by using the wizard or by choosing the **Syntax** tab and specifying your rule syntax manually.

To continue using the wizard, do the following:

- a. For **Log group(s)**, select the log groups that you want the rule to monitor. You can select as many as 20 log groups.

To select all log groups with names that start with a certain string, choose **Select by prefix match** and enter that prefix.

If you use **Select by prefix match**, be aware of how many log groups will match your prefix and be analyzed by the rule. You incur charges for each log event that matches a rule. If you accidentally search more log groups than you intend, you might incur unexpected charges. For more information, see [Amazon CloudWatch Pricing](#).

- b. For **Log format**, choose **JSON** or **CLF**.
- c. For **Contribution, Key**, enter a contributor type that you want to report on. The report displays the top-N values for this contributor type.

Valid entries are any log field that has values. Examples include **requestId**, **sourceIPAddress**, and **containerID**.

For information about finding the log field names for the logs in a certain log group, see [Finding Log Fields \(p. 146\)](#).

Keys larger than 1 KB are truncated to 1KB.

- d. (Optional) Add more values for **Contribution, Key**. You can include as many as four keys in a rule. If you enter more than one key, the contributors in the report are defined by unique value combinations of the keys. For example, if you specify three keys, each unique combination of values for the three keys is counted as a unique contributor.
- e. (Optional) If you want to rank contributors by the value of a numerical log field, instead of by number of occurrences in log events, use the **Contribution, Value** field. Enter the name of a numerical log field that you want to sum to determine contributor ranking. For example, if you want to find the source IP addresses that are sending the most bytes over the network, you would add **bytes** as the value, assuming that **bytes** is the correct keyword for that field in the log events.
- f. If you want to filter the results to a narrower scope, choose **Add filter**. For **Match**, enter the log field that you want to filter by. For **Condition**, choose the comparison operator and enter a value that you want to filter this field for. You can use ***** as a wildcard in the value.

For example, if you want to analyze only the log events in an Apache log that contain errors, for **Match**, you would specify **RESPONSE_CODE**, for **Condition**, you would specify **EqualTo**, and then you would enter **5**** as the value to filter for.

You can add as many as 10 filters in a rule. Multiple filters are joined by AND logic, so only log events that match all filters are evaluated.

- g. For **Aggregate on**, choose **COUNT** or **SUM**. Choosing **COUNT** causes the contributor ranking to be based on the number of occurrences. Choosing **SUM** causes the ranking to be based on the aggregated sum of the values of the field that you specify for **Contribution, Value**.
7. To enter your rule as a JSON object instead of using the wizard, do the following:
 - a. Choose the **Syntax** tab.
 - b. In **Rule body**, enter the JSON object for your rule. For information about rule syntax, see [Contributor Insights Rule Syntax \(p. 147\)](#).
 8. Choose whether to create the rule in a disabled or enabled state. If you choose to enable it, the rule immediately starts analyzing your data. You incur costs when you run enabled rules. For more information, see [Amazon CloudWatch Pricing](#).

Contributor Insights analyzes only new log events after a rule is created. A rule cannot process logs events that were previously processed by CloudWatch Logs.

9. Choose **Create**.

You can disable, enable, or delete rules that you have created.

To enable, disable, or delete a rule in Contributor Insights

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Contributor Insights**.
3. In the list of rules, select the check box next to a single rule.

Built-in rules are created by AWS services and can't be edited, disabled, or deleted.

4. Choose **Actions**, and then choose the option you want.

Finding Log Fields

When you create a rule, you need to know the names of fields in the log entries in a log group.

To find the log fields in a log group

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

2. In the navigation pane, under **Logs**, choose **Insights**.
3. Above the query editor, select one or more log groups to query.

When you select a log group, CloudWatch Logs Insights automatically detects fields in the data in the log group and displays them in the right pane in **Discovered fields**.

Contributor Insights Rule Syntax

This section explains the syntax for Contributor Insights rules. Use this syntax only when you are creating a rule by entering a JSON block. If you use the wizard to create a rule, you don't need to know the syntax. For more information about creating rules using the wizard, see [Creating a Contributor Insights Rule \(p. 144\)](#).

All matching of rules to log event field names and values is case sensitive.

The following example illustrates the syntax for JSON logs.

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "API-Gateway-Access-Logs*",
    "Log-group-name2"
  ],
  "LogFormat": "JSON",
  "Contribution": {
    "Keys": [
      "$.ip"
    ],
    "ValueOf": "$.requestBytes",
    "Filters": [
      {
        "Match": "$.httpMethod",
        "In": [
          "PUT"
        ]
      }
    ]
  },
  "AggregateOn": "Sum"
}
```

Fields in Contributor Insights Rules

Schema

The value of Schema for a rule that analyzes CloudWatch Logs data must always be { "Name" : "CloudWatchLogRule", "Version": 1 }

LogGroupNames

An array of strings. For each element in the array, you can optionally use * at the end of a string to include all log groups with names that start with that prefix.

Be careful about using wildcards with log group names. You incur charges on each log event that matches a rule. If you accidentally search more log groups than you intend, you might incur unexpected charges. For more information, see [Amazon CloudWatch Pricing](#).

LogFormat

Valid values are `JSON` and `CLF`.

Contribution

This object includes a `Keys` array with as many as four members, optionally a single `ValueOf`, and optionally an array of as many as four `Filters`.

Keys

An array of up to four log fields that are used as dimensions to classify contributors. If you enter more than one key, each unique combination of values for the keys is counted as a unique contributor. The fields must be specified using JSON property format notation.

ValueOf

(Optional) Specify this only when you are specifying `SUM` as the value of `AggregateOn`. `ValueOf` specifies a log field with numerical values. In this type of rule, the contributors are ranked by their sum of the value of this field, instead of their number of occurrences in the log entries. For example, if you want to sort contributors by their total `BytesSent` over a period, you would set `ValueOf` to `BytesSent` and specify `Sum` for `AggregateOn`.

Filters

(Optional) Specifies an array of as many as four filters to narrow the log events that are included in the report. If you specify multiple filters, Contributor Insights evaluates them with a logical AND operator. You can use this to filter out irrelevant log events in your search or you can use it to select a single contributor to analyze their behavior.

Each member in the array must include a `Match` field and a field indicating the type of matching operator to use.

The `Match` field specifies a log field to evaluate in the filter. The log field is specified using JSON property format notation.

The matching operator field must be one of the following: `In`, `NotIn`, `StartsWith`, `GreaterThan`, `LessThan`, `EqualTo`, `NotEqualTo`, or `IsPresent`. If the operator field is `In`, `NotIn`, or `StartsWith`, it is followed by an array of string values to check for. Contributor Insights evaluates the array of string values with an OR operator. The array can include as many as 10 string values.

If the operator field is `GreaterThan`, `LessThan`, `EqualTo`, or `NotEqualTo`, it is followed by a single numerical value to compare with.

If the operator field is `IsPresent`, it is followed by either `true` or `false`. This operator matches log events based on whether the specified log field is present in the log event. The `isPresent` works only with values in the leaf node of JSON properties. For example, a filter that looks for matches to `c-count` does not evaluate a log event with a value of `details.c-count.c1`.

See the following for filter examples:

```
{ "Match": "$.httpMethod", "In": [ "PUT", ] }
{ "Match": "$.StatusCode", "EqualTo": 200 }
{ "Match": "$.BytesReceived", "GreaterThan": 10000 }
{ "Match": "$.eventSource", "StartsWith": [ "ec2", "ecs" ] }
```

AggregateOn

Valid values are `COUNT` and `SUM`. Specifies whether to aggregate the report based on a count of occurrences or a sum of the values of the field that is specified in the `ValueOf` field.

JSON Property Format Notation

The `Keys`, `ValueOf`, and `Match` fields follow JSON property format with dot notation, where `$` represents the root of the JSON object. This is followed by a period and then an alphanumeric string with the name of the subproperty. Multiple property levels are supported.

The following list illustrates valid examples of JSON property format:

```
$.userAgent
$.endpoints[0]
$.users[1].name
$.requestParameters.instanceId
```

Additional Field in Rules for CLF Logs

Common Log Format (CLF) log events do not have names for the fields like JSON does. To provide the fields to use for Contributor Insights rules, a CLF log event can be treated as array with an index starting from 1. You can specify the first field as `"1"`, the second field as `"2"`, and so on.

To make a rule for a CLF log easier to read, you can use `Fields`. This enables you to provide a naming alias for CLF field locations. For example, you can specify that the location `"4"` is an IP address. Once specified, `IpAddress` can be used as property in the `Keys`, `ValueOf`, and `Filters` in the rule.

The following is an example of a rule for a CLF log that uses the `Fields` field.

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "API-Gateway-Access-Logs*"
  ],
  "LogFormat": "CLF",
  "Fields": {
    "4": "IpAddress",
    "7": "StatusCode"
  },
  "Contribution": {
    "Keys": [
      "IpAddress"
    ],
    "Filters": [
      {
        "Match": "StatusCode",
        "EqualTo": 200
      }
    ]
  },
  "AggregateOn": "Count"
}
```

Contributor Insights Rule Examples

This section contains examples that illustrate use cases for Contributor Insights rules.

VPC Flow Logs: Byte Transfers by Source and Destination IP Address

```
{
  "Schema": {
```

```
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "/aws/containerinsights/sample-cluster-name/flowlogs"
  ],
  "LogFormat": "CLF",
  "Fields": {
    "4": "srcaddr",
    "5": "dstaddr",
    "10": "bytes"
  },
  "Contribution": {
    "Keys": [
      "srcaddr",
      "dstaddr"
    ],
    "ValueOf": "bytes",
    "Filters": []
  },
  "AggregateOn": "Sum"
}
```

VPC Flow Logs: Highest Number of HTTPS Requests

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "/aws/containerinsights/sample-cluster-name/flowlogs"
  ],
  "LogFormat": "CLF",
  "Fields": {
    "5": "destination address",
    "7": "destination port",
    "9": "packet count"
  },
  "Contribution": {
    "Keys": [
      "destination address"
    ],
    "ValueOf": "packet count",
    "Filters": [
      {
        "Match": "destination port",
        "EqualTo": 443
      }
    ]
  },
  "AggregateOn": "Sum"
}
```

VPC Flow Logs: Rejected TCP Connections

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "/aws/containerinsights/sample-cluster-name/flowlogs"
  ]
}
```

```
],
"LogFormat": "CLF",
"Fields": {
  "3": "interfaceID",
  "4": "sourceAddress",
  "8": "protocol",
  "13": "action"
},
"Contribution": {
  "Keys": [
    "interfaceID",
    "sourceAddress"
  ],
  "Filters": [
    {
      "Match": "protocol",
      "EqualTo": 6
    },
    {
      "Match": "action",
      "In": [
        "REJECT"
      ]
    }
  ]
},
"AggregateOn": "Sum"
}
```

Viewing Contributor Insights Reports

To view graphs of report data and a ranked list of contributors found by your rules, follow these steps.

To view your rule reports

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Contributor Insights**.
3. In the list of rules, choose the name of a rule.

The graph displays the results of the rule over the last three hours. The table under the graph shows the top 10 contributors.

4. To change the number of contributors shown in the table, choose **Top 10 contributors** at the top of the graph.
5. To filter the graph to show only the results from a single contributor, choose that contributor in the table legend. To again show all contributors, choose that same contributor again in the legend.
6. To change the time range shown in the report, choose **15m**, **30m**, **1h**, **2h**, **3h**, or **custom** at the top of the graph.

The maximum time range for the report is 24 hours, but you can choose a 24-hour window that occurred up to 15 days ago. To choose a time window in the past, choose **custom**, **absolute**, and then specify your time window.

7. To change the length of the time period used for the aggregation and ranking of contributors, choose **period** at the top of the graph. Viewing a longer time period generally shows a smoother report with few spikes. Choosing a shorter time period is more likely to display spikes.
8. To add this graph to a CloudWatch dashboard, choose **Add to dashboard**.
9. To open the CloudWatch Logs Insights query window, with the log groups in this report already loaded in the query box, choose **View logs**.

10. To export the report data to your clipboard or a CSV file, choose **Export**.

Graphing Metrics Generated by Rules

Contributor Insights provides a metric math function, `INSIGHT_RULE_METRIC`. You can use this function to add data from a Contributor Insights report to a graph in the **Metrics** tab of the CloudWatch console. You can also set an alarm based on this math function. For more information about metric math functions, see [Using Metric Math \(p. 52\)](#)

To use this metric math function, you must be signed in to an account that has both the `cloudwatch:GetMetricData` and `cloudwatch:GetInsightRuleReport` permissions.

The syntax is `INSIGHT_RULE_METRIC(ruleName, metricName)`. *ruleName* is the name of a Contributor Insights rule. *metricName* is one of the values in the following list. The value of *metricName* determines which type of data the math function returns.

- `UniqueContributors` — the number of unique contributors for each data point.
- `MaxContributorValue` — the value of the top contributor for each data point. The identity of the contributor might change for each data point in the graph.

If this rule aggregates by `COUNT`, the top contributor for each data point is the contributor with the most occurrences in that period. If the rule aggregates by `SUM`, the top contributor is the contributor with the greatest sum in the log field specified by the rule's `value` during that period.

- `SampleCount` — the number of data points matched by the rule.
- `Sum` — the sum of the values from all contributors during the time period represented by that data point.
- `Minimum` — the minimum value from a single observation during the time period represented by that data point.
- `Maximum` — the maximum value from a single observation during the time period represented by that data point.
- `Average` — the average value from all contributors during the time period represented by that data point.

Setting an Alarm on Contributor Insights Metric Data

You can set alarms on metrics generated by Contributor Insights by using `INSIGHT_RULE_METRIC`. For example, you can create an alarm based on the percentage of TCP connections that have been rejected. First, create two rules like the following:

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "/aws/containerinsights/sample-cluster-name/flowlogs"
  ],
  "LogFormat": "CLF",
  "Fields": {
    "3": "interfaceID",
    "4": "sourceAddress",
    "8": "protocol",
    "13": "action"
  },
}
```



```
"Contribution": {
  "Keys": [
    "interfaceID",
    "sourceAddress"
  ],
  "Filters": [
    {
      "Match": "protocol",
      "EqualTo": 6
    },
    {
      "Match": "action",
      "In": [
        "REJECT"
      ]
    }
  ]
},
"AggregateOn": "Sum"
}
```

```
{
  "Schema": {
    "Name": "CloudWatchLogRule",
    "Version": 1
  },
  "LogGroupNames": [
    "/aws/containerinsights/sample-cluster-name/flowlogs"
  ],
  "LogFormat": "CLF",
  "Fields": {
    "3": "interfaceID",
    "4": "sourceAddress",
    "8": "protocol",
    "13": "action"
  },
  "Contribution": {
    "Keys": [
      "interfaceID",
      "sourceAddress"
    ],
    "Filters": [
      {
        "Match": "protocol",
        "EqualTo": 6
      }
    ]
  },
  "AggregateOn": "Sum"
}
```

Then, in the **Metrics** tab on the console, create a graph with the following metric math expressions:

```
e1 INSIGHT_RULE_METRIC("RejectedConnectionsRule", "Sum")
e2 INSIGHT_RULE_METRIC("TotalConnectionsRule", "Sum")
e3 (e1/e2)*100
```

For more information about graphing metrics and using metric math functions, see [Adding a Math Expression to a CloudWatch Graph \(p. 53\)](#).

In this example, the e3 expression returns the percentage of connections that are rejected. If you want to be notified when 20 percent of connections are rejected, you can set an alarm on that expression, setting **20** as the threshold. To set an alarm on a metric you are viewing in the **Metrics** tab, choose the alarm icon in the row of the metric that you want to alarm on. The alarm icon looks like a bell.

For more information about graphing metrics and using metric math functions, see [Adding a Math Expression to a CloudWatch Graph \(p. 53\)](#).

Using Contributor Insights Built-In Rules

Other AWS services create built-in Contributor Insights rules that evaluate metrics from those AWS services. Currently, Amazon DynamoDB supports built-in rules.

For more information, see [Contributor Insights for Amazon DynamoDB](#).

Using Container Insights

Use CloudWatch Container Insights to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. Container Insights is available for Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and Kubernetes platforms on Amazon EC2. Amazon ECS support includes support for Fargate.

The metrics include utilization for resources such as CPU, memory, disk, and network. Container Insights also provides diagnostic information, such as container restart failures, to help you isolate issues and resolve them quickly. You can also set CloudWatch alarms on metrics that Container Insights collects.

The metrics that Container Insights collects are available in CloudWatch automatic dashboards. You can analyze and troubleshoot container performance and logs data with CloudWatch Logs Insights.

Operational data is collected as *performance log events*. These are entries that use a structured JSON schema that enables high-cardinality data to be ingested and stored at scale. From this data, CloudWatch creates aggregated metrics at the cluster, node, pod, task, and service level as CloudWatch metrics.

Metrics collected by Container Insights are charged as custom metrics. For more information about CloudWatch pricing, see [Amazon CloudWatch Pricing](#).

In Amazon EKS and Kubernetes, Container Insights uses a containerized version of the CloudWatch agent to discover all of the running containers in a cluster. It then collects performance data at every layer of the performance stack.

Container Insights supports encryption with the customer master key (CMK) for the logs and metrics that it collects. To enable this encryption, you must manually enable KMS encryption for the log group that receives Container Insights data. This results in Container Insights encrypting this data using the provided CMK. Only symmetric CMKs are supported. Do not use asymmetric CMKs to encrypt your log groups.

For more information, see [Encrypt Log Data in CloudWatch Logs Using AWS KMS](#).

Supported Platforms

Container Insights is available for Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, and Kubernetes platforms on Amazon EC2 instances.

- For Amazon ECS, Container Insights collects metrics at the cluster, task and service levels on both Linux and Windows Server instances. It can collect metrics at the instance-level only on Linux instances.

For Amazon ECS, network metrics are available only for containers in `bridge` network mode. They are not available for containers in `awsvpc` network mode or `host` network mode.

- For Amazon Elastic Kubernetes Service, and Kubernetes platforms on Amazon EC2 instances, Container Insights is supported only on Linux instances.
- Currently, Container Insights isn't supported in AWS Batch.

Supported Regions for Amazon ECS

Container Insights for Amazon ECS is supported in the following Regions:

- US East (N. Virginia)

- US East (Ohio)
- US West (N. California)
- US West (Oregon)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- Asia Pacific (Hong Kong)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Mumbai)
- South America (São Paulo)

AWS Fargate is not supported in Europe (Paris) or South America (São Paulo).

Supported Regions for Amazon EKS and Kubernetes

Container Insights for Amazon EKS and Kubernetes is supported in the following Regions:

- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- South America (São Paulo)

Topics

- [Setting Up Container Insights \(p. 157\)](#)
- [Viewing Container Insights Metrics \(p. 176\)](#)
- [Metrics Collected by Container Insights \(p. 178\)](#)
- [Container Insights Performance Log Reference \(p. 185\)](#)
- [Container Insights Prometheus Metrics Monitoring \(p. 205\)](#)

- [Troubleshooting Container Insights \(p. 233\)](#)
- [Building Your Own CloudWatch Agent Docker Image \(p. 234\)](#)
- [Deploying Other CloudWatch Agent Features in Your Containers \(p. 234\)](#)

Setting Up Container Insights

The Container Insights setup process is different for Amazon ECS and Amazon EKS and Kubernetes.

Topics

- [Setting Up Container Insights on Amazon ECS \(p. 157\)](#)
- [Setting Up Container Insights on Amazon EKS and Kubernetes \(p. 164\)](#)

Setting Up Container Insights on Amazon ECS

You can use one or both of the following options to enable Container Insights on Amazon ECS clusters:

- Use the AWS Management Console or the AWS CLI to start collecting cluster-level, task-level, and service-level metrics.
- Deploy the CloudWatch agent as a DaemonSet to start collecting of instance-level metrics on clusters that are hosted on Amazon EC2 instances.

Topics

- [Setting Up Container Insights on Amazon ECS for Cluster- and Service-Level Metrics \(p. 157\)](#)
- [Deploying the CloudWatch Agent to Collect EC2 Instance-Level Metrics on Amazon ECS \(p. 159\)](#)

Setting Up Container Insights on Amazon ECS for Cluster- and Service-Level Metrics

You can enable Container Insights on new and existing Amazon ECS clusters. Container Insights collects metrics at the cluster, task, and service levels. For existing clusters, you use the AWS CLI. For new clusters, use either the Amazon ECS console or the AWS CLI.

If you're using Amazon ECS on an Amazon EC2 instance, and you want to collect network and storage metrics from Container Insights, launch that instance using an AMI that includes Amazon ECS agent version 1.29. For information about updating your agent version, see [Updating the Amazon ECS Container Agent](#)

You can use the AWS CLI to set account-level permission to enable Container Insights for any new Amazon ECS clusters created in your account. To do so, enter the following command.

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

Setting Up Container Insights on Existing Amazon ECS Clusters

To enable Container Insights on an existing Amazon ECS cluster, enter the following command. You must be running version 1.16.200 or later of the AWS CLI for the following command to work.

```
aws ecs update-cluster-settings --cluster myECScluster --settings  
name=containerInsights,value=enabled
```

Setting Up Container Insights on New Amazon ECS Clusters

There are two ways to enable Container Insights on new Amazon ECS clusters. You can configure Amazon ECS so that all new clusters are enabled for Container Insights by default. Otherwise, you can enable a new cluster when you create it.

Using the AWS Management Console

You can enable Container Insights on all new clusters by default, or on an individual cluster as you create it.

To enable Container Insights on all new clusters by default

1. Open the Amazon ECS console at <https://console.aws.amazon.com/ecs/>.
2. In the navigation pane, choose **Account Settings**.
3. Select the check box at the bottom of the page to enable the Container Insights default.

If you haven't used the preceding procedure to enable Container Insights on all new clusters by default, use the following steps to create a cluster with Container Insights enabled.

To create a cluster with Container Insights enabled

1. Open the Amazon ECS console at <https://console.aws.amazon.com/ecs/>.
2. In the navigation pane, choose **Clusters**.
3. Choose **Create cluster**.
4. On the next page, do the following:
 - a. Name your cluster.
 - b. If you don't have a VPC already, select the check box to create one. You can use the default values for the VPC.
 - c. Fill out all other needed information, including instance type.
 - d. Select **Enabled Container Insights**.
 - e. Choose **Create**.

You can now create task definitions, run tasks, and launch services in the cluster. For more information, see the following:

- [Creating a Task Definition](#)
- [Running Tasks](#)
- [Creating a Service](#)

Setting Up Container Insights on New Amazon ECS Clusters Using the AWS CLI

To enable Container Insights on all new clusters by default, enter the following command.

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

If you didn't use the preceding command to enable Container Insights on all new clusters by default, enter the following command to create a new cluster with Container Insights enabled. You must be running version 1.16.200 or later of the AWS CLI for the following command to work.

```
aws ecs create-cluster --cluster-name myCICluster --settings  
"name=containerInsights,value=enabled"
```

Disabling Container Insights on Amazon ECS Clusters

To disable Container Insights on an existing Amazon ECS cluster, enter the following command.

```
aws ecs update-cluster-settings --cluster myECICluster --settings  
name=containerInsights,value=disabled
```

Deploying the CloudWatch Agent to Collect EC2 Instance-Level Metrics on Amazon ECS

To deploy the CloudWatch agent to collect instance-level metrics from Amazon ECS clusters that are hosted on EC2 instance, use a quick start setup with a default configuration, or install the agent manually to be able to customize it.

Both methods require that you already have at least one Amazon ECS cluster deployed with an EC2 launch type. These methods also assume that you have the AWS CLI installed. Additionally, to run the commands in the following procedures, you must be logged on to an account or role that has the **IAMFullAccess** and **AmazonECS_FullAccess** policies.

Topics

- [Quick Setup Using AWS CloudFormation \(p. 159\)](#)
- [Manual and Custom Setup \(p. 160\)](#)

Quick Setup Using AWS CloudFormation

To use the quick setup, enter the following command to use AWS CloudFormation to install the agent. Replace *cluster-name* and *cluster-region* with the name and Region of your Amazon ECS cluster.

This command creates the IAM roles **CWAgentECSTaskRole** and **CWAgentECSExecutionRole**. If these roles already exist in your account, use `ParameterKey=CreateIAMRoles,ParameterValue=False` instead of `ParameterKey=CreateIAMRoles,ParameterValue=True` when you enter the command. Otherwise, the command will fail.

```
ClusterName=cluster-name  
Region=cluster-region  
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \  
--template-body https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-  
container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/  
cwagent-ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json \  
--parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \  
ParameterKey=CreateIAMRoles,ParameterValue=True \  
--capabilities CAPABILITY_NAMED_IAM \  
--region ${Region}
```

(Alternative) Using Your Own IAM Roles

If you want to use your own custom ECS task role and ECS task execution role instead of the **CWAgentECSTaskRole** and **CWAgentECSExecutionRole** roles, first make sure that the role to be used as the ECS task role has **CloudWatchAgentServerPolicy** attached. Also, make sure that the role to be used as the ECS task execution role has both the **CloudWatchAgentServerPolicy** and **AmazonECSTaskExecutionRolePolicy** policies attached. Then enter the following command. In the command, replace *task-role-arn* with the ARN of your custom ECS task role, and replace *execution-role-arn* with the ARN of your custom ECS task execution role.

```
ClusterName=cluster-name  
Region=cluster-region
```

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
  --template-body https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
  container-insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/
  cwagent-ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json \
  --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
    ParameterKey=TaskRoleArn,ParameterValue=${TaskRoleArn} \
    ParameterKey=ExecutionRoleArn,ParameterValue=${ExecutionRoleArn} \
  --capabilities CAPABILITY_NAMED_IAM \
  --region ${Region}
```

Troubleshooting the Quick Setup

To check the status of the AWS CloudFormation stack, enter the following command.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stacks --stack-name CWAgentECS-${ClusterName}-${Region} --region
  $Region
```

If you see the StackStatus is other than CREATE_COMPLETE or CREATE_IN_PROGRESS, check the stack events to find the error. Enter the following command.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name CWAgentECS-${ClusterName}-${Region} --
  region $Region
```

To check the status of the cwagent daemon service, enter the following command. In the output, you should see that the runningCount is equal to the desiredCount in the deployment section. If it isn't equal, check the failures section in the output.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services cwagent-daemon-service --cluster $ClusterName --region
  $Region
```

You can also use the CloudWatch Logs console to check the agent log. Look for the **/ecs/ecs-cwagent-daemon-service** log group.

Deleting the AWS CloudFormation Stack for the CloudWatch Agent

If you need to delete the AWS CloudFormation stack, enter the following command.

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation delete-stack --stack-name CWAgentECS-${ClusterName}-${Region} --region
  ${Region}
```

Manual and Custom Setup

Follow the steps in this section to manually deploy the CloudWatch agent to collect instance-level metrics from your Amazon ECS clusters that are hosted on EC2 instances.

Necessary IAM Roles and Policies

Two IAM roles are required. You must create them if they don't already exist. For more information about these roles, see [Amazon ECS Task Role](#) and [Amazon ECS Task Execution Role](#).

- An *ECS task role*, which is used by the CloudWatch agent to publish metrics. If this role already exists, you must make sure it has the `CloudWatchAgentServerPolicy` policy attached.
- An *ECS task execution role*, which is used by Amazon ECS agent to launch the CloudWatch agent. If this role already exists, you must make sure it has the `AmazonECSTaskExecutionRolePolicy` and `CloudWatchAgentServerPolicy` policies attached.

If you do not already have these roles, you can use the following commands to create them and attach the necessary policies. This first command creates the ECS task role.

```
aws iam create-role --role-name CWAgentECSTaskRole \
  --assume-role-policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com\"}, \"Action\": \"sts:AssumeRole\"}]}"
```

After you enter the previous command, note the value of `Arn` from the command output as `"TaskRoleArn"`. You'll need to use it later when you create the task definition. Then enter the following command to attach the necessary policies.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-name CWAgentECSTaskRole
```

This next command creates the ECS task execution role.

```
aws iam create-role --role-name CWAgentECSExecutionRole \
  --assume-role-policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com\"}, \"Action\": \"sts:AssumeRole\"}]}"
```

After you enter the previous command, note the value of `Arn` from the command output as `"ExecutionRoleArn"`. You'll need to use it later when you create the task definition. Then enter the following commands to attach the necessary policies.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-name CWAgentECSExecutionRole

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy \
  --role-name CWAgentECSExecutionRole
```

Create the Task Definition and Launch the Daemon Service

Create a task definition and use it to launch the CloudWatch agent as a daemon service. To create the task definition, enter the following command. In the first lines, replace the placeholders with the actual values for your deployment. *logs-region* is the Region where CloudWatch Logs is located, and *cluster-region* is the Region where your cluster is located. *task-role-arn* is the Arn of the ECS task role that you are using, and *execution-role-arn* is the Arn of the ECS task execution role.

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-ecs-instance-
metric/cwagent-ecs-instance-metric.json \
```

```
| sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|  
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \  
| xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

Then run the following command to launch the daemon service. Replace *cluster-name* and *cluster-region* with the name and Region of your Amazon ECS cluster.

```
ClusterName=cluster-name  
Region=cluster-region  
aws ecs create-service \  
    --cluster ${ClusterName} \  
    --service-name cwagent-daemon-service \  
    --task-definition ecs-cwagent-daemon-service \  
    --scheduling-strategy DAEMON \  
    --region ${Region}
```

If you see this error message, An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent, you have already created a daemon service named cwagent-daemon-service. You must delete that service first, using the following command as an example.

```
ClusterName=cluster-name  
Region=cluster-region  
aws ecs delete-service \  
    --cluster ${ClusterName} \  
    --service cwagent-daemon-service \  
    --region ${Region} \  
    --force
```

(Optional) Advanced Configuration

Optionally, you can use SSM to specify other configuration options for the CloudWatch agent in your Amazon ECS clusters that are hosted on EC2 instances. These options are as follows:

- `metrics_collection_interval` – How often in seconds that the CloudWatch agent collects metrics. The default is 60. The range is 1–172,000.
- `endpoint_override` – (Optional) Specifies a different endpoint to send logs to. You might want to do this if you're publishing from a cluster in a VPC and you want the logs data to go to a VPC endpoint.

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

- `force_flush_interval` – Specifies in seconds the maximum amount of time that logs remain in the memory buffer before being sent to the server. No matter the setting for this field, if the size of the logs in the buffer reaches 1 MB, the logs are immediately sent to the server. The default value is 5 seconds.
- `region` – By default, the agent publishes metrics to the same Region where the Amazon ECS container instance is located. To override this, you can specify a different Region here. For example, `"region" : "us-east-1"`

The following is an example of a customized configuration:

```
{  
  "agent": {  
    "region": "us-east-1"  
  },  
  "logs": {  
    "metrics_collected": {  
      "ecs": {  
        "metrics_collection_interval": 30  
      }  
    }  
  }  
}
```

```
    },  
    "force_flush_interval": 5  
  }  
}
```

To customize your CloudWatch agent configuration in your Amazon ECS containers

1. Make sure that the **AmazonSSMReadOnlyAccess** policy is attached to your Amazon ECS Task Execution role. You can enter the following command to do so. This example assumes that your Amazon ECS Task Execution role is **CWAgentECSExecutionRole**. If you are using a different role, substitute that role name in the following command.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess \br/>    --role-name CWAgentECSExecutionRole
```

2. Create the customized configuration file similar to the preceding example. Name this file `/tmp/ecs-cwagent-daemon-config.json`.
3. Run the following command to put this configuration into the Parameter Store. Replace **cluster-region** with the Region of your Amazon ECS cluster. To run this command, you must be logged on to a user or role that has the **AmazonSSMFullAccess** policy.

```
Region=cluster-region  
aws ssm put-parameter \br/>    --name "ecs-cwagent-daemon-service" \br/>    --type "String" \br/>    --value "`cat /tmp/ecs-cwagent-daemon-config.json`" \br/>    --region $Region
```

4. Download the task definition file to a local file, such as `/tmp/cwagent-ecs-instance-metric.json`

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/  
latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-ecs-  
instance-metric/cwagent-ecs-instance-metric.json -o /tmp/cwagent-ecs-instance-  
metric.json
```

5. Modify the task definition file. Remove the following section:

```
"environment": [  
    {  
        "name": "USE_DEFAULT_CONFIG",  
        "value": "True"  
    }  
],
```

Replace that section with the following:

```
"secrets": [  
    {  
        "name": "CW_CONFIG_CONTENT",  
        "valueFrom": "ecs-cwagent-daemon-service"  
    }  
],
```

6. Restart the agent as a daemon service by following these steps:
 - a. Run the following command.

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
cat /tmp/cwagent-ecs-instance-metric.json \
| sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
| xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

- b. Run the following command to launch the daemon service. Replace *cluster-name* and *cluster-region* with the name and Region of your Amazon ECS cluster.

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
  --cluster ${ClusterName} \
  --service-name cwagent-daemon-service \
  --task-definition ecs-cwagent-daemon-service \
  --scheduling-strategy DAEMON \
  --region ${Region}
```

If you see this error message, An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent, you have already created a daemon service named cwagent-daemon-service. You must delete that service first, using the following command as an example.

```
ClusterName=cluster-name
Region=Region
aws ecs delete-service \
  --cluster ${ClusterName} \
  --service cwagent-daemon-service \
  --region ${Region} \
  --force
```

Setting Up Container Insights on Amazon EKS and Kubernetes

The overall process for setting up Container Insights on Amazon EKS or Kubernetes is as follows:

1. Verify that you have the necessary prerequisites.
2. Set up the CloudWatch agent as a DaemonSet on your Amazon EKS cluster or Kubernetes cluster to send metrics to CloudWatch, and set up FluentD as a DaemonSet to send logs to CloudWatch Logs.

You can perform these steps at once as part of the quick start setup, or do them separately.

3. (Optional) Set up Amazon EKS control plane logging.
4. (Optional) Set up the CloudWatch agent as a StatsD endpoint on the cluster to send StatsD metrics to CloudWatch.
5. (Optional) Enable App Mesh Envoy Access Logs.

Topics

- [Verify Prerequisites \(p. 165\)](#)
- [Quick Start Setup for Container Insights on Amazon EKS \(p. 166\)](#)
- [Set Up the CloudWatch Agent to Collect Cluster Metrics \(p. 166\)](#)

- [Set Up FluentD as a DaemonSet to Send Logs to CloudWatch Logs \(p. 170\)](#)
- [\(Optional\) Set Up Amazon EKS Control Plane Logging \(p. 174\)](#)
- [\(Optional\) Enable App Mesh Envoy Access Logs \(p. 174\)](#)
- [Updating or Deleting Container Insights on Amazon EKS and Kubernetes \(p. 175\)](#)

Verify Prerequisites

Before you install Container Insights on Amazon EKS or Kubernetes, verify the following:

- You have a functional Amazon EKS or Kubernetes cluster with nodes attached in one of the Regions that supports the Container Insights for Amazon EKS and Kubernetes. For the list of supported Regions, see [Using Container Insights \(p. 155\)](#).
- You have `kubectl` installed and running. For more information, see [Installing kubectl](#) in the *Amazon EKS User Guide*.
- If you're using Kubernetes running on AWS instead of using Amazon EKS, the following prerequisites are also necessary:
 - Be sure that your Kubernetes cluster has enabled role-based access control (RBAC). For more information, see [Using RBAC Authorization](#) in the Kubernetes Reference.
 - Your kubelet has enabled Webhook authorization mode. For more information, see [Kubelet authentication/authorization](#) in the Kubernetes Reference.
 - Your container runtime is Docker.

You must also attach a policy to the IAM role of your Amazon EKS worker nodes to enable them to send metrics and logs to CloudWatch.

To add the necessary policy to the IAM role for your worker nodes

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Select one of the worker node instances and choose the IAM role in the description.
3. On the IAM role page, choose **Attach policies**.
4. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. If necessary, use the search box to find this policy.
5. Choose **Attach policies**.

If you're running a Kubernetes cluster outside Amazon EKS, you might not already have an IAM role attached to your worker nodes. If not, you must first attach an IAM role to the instance and then add the policy as explained in the previous steps. For more information on attaching a role to an instance, see [Attaching an IAM Role to an Instance](#) in the *Amazon EC2 User Guide for Windows Instances*.

If you're running a Kubernetes cluster outside Amazon EKS and you want to collect EBS volume IDs in the metrics, you must add another policy to the IAM role attached to the instance. Add the following as an inline policy. For more information, see [Adding and Removing IAM Identity Permissions](#) in the *IAM User Guide*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeVolumes"
      ],
      "Resource": "*",
    }
  ]
}
```

```
        "Effect": "Allow"
      }
    ]
  }
}
```

Quick Start Setup for Container Insights on Amazon EKS

To complete the setup of Container Insights, you can follow the quick start instructions in this section.

Alternatively, you can instead follow the instructions in the following two sections, [Set Up the CloudWatch Agent to Collect Cluster Metrics \(p. 166\)](#) and [Set Up FluentD as a DaemonSet to Send Logs to CloudWatch Logs \(p. 170\)](#). Those sections provide more details on how CloudWatch agent works with Amazon EKS and the configuration, but require you to perform more installation steps.

To deploy Container Insights using the quick start, enter the following command.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-
monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/cluster-
name/;s/{{region_name}}/cluster-region/" | kubectl apply -f -
```

In this command, **cluster-name** is the name of your Amazon EKS or Kubernetes cluster, and **cluster-region** is the name of the Region where the logs are published. We recommend that you use the same Region where your cluster is deployed to reduce the AWS outbound data transfer costs.

For example, to deploy Container Insights on the cluster named `MyCluster` and publish the logs and metrics to US West (Oregon), enter the following command.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-
monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/
MyCluster/;s/{{region_name}}/us-west-2/" | kubectl apply -f -
```

Deleting Container Insights

If you want to remove Container Insights after using the quick start setup, enter the following command.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/
latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-
monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/cluster-
name/;s/{{region_name}}/cluster-region/" | kubectl delete -f -
```

Set Up the CloudWatch Agent to Collect Cluster Metrics

To set up Container Insights to collect metrics, you can follow the steps in [Quick Start Setup for Container Insights on Amazon EKS \(p. 166\)](#) or you can follow the steps in this section. In the following steps, you set up the CloudWatch agent to be able to collect metrics from your clusters.

Step 1: Create a Namespace for CloudWatch

Use the following step to create a Kubernetes namespace called `amazon-cloudwatch` for CloudWatch. You can skip this step if you have already created this namespace.

To create a namespace for CloudWatch

- Enter the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

Step 2: Create a Service Account in the Cluster

Use the following step to create a service account for the CloudWatch agent, if you do not already have one.

To create a service account for the CloudWatch agent

- Enter the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

If you didn't follow the previous steps, but you already have a service account for the CloudWatch agent that you want to use, you must ensure that it has the following rules. Additionally, in the rest of the steps in the Container Insights installation, you must use the name of that service account instead of `cloudwatch-agent`.

```
rules:
- apiGroups: [""]
  resources: ["pods", "nodes", "endpoints"]
  verbs: ["watch", "list"]
- apiGroups: [""]
  resources: ["nodes/proxy"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["nodes/stats", "configmaps", "events"]
  verbs: ["create"]
- apiGroups: [""]
  resources: ["configmaps"]
  resourceNames: ["cwagent-clusterleader"]
  verbs: ["get", "update"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
```

Step 3: Create a ConfigMap for the CloudWatch Agent

Use the following steps to create a ConfigMap for the CloudWatch agent.

To create a ConfigMap for the CloudWatch agent

1. Download the ConfigMap YAML to your `kubectl` client host by running the following command:

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-configmap.yaml
```

2. Edit the downloaded YAML file, as follows:

- **cluster_name** – In the `kubernetes` section, replace `{{cluster-name}}` with the name of your cluster. Remove the `{{}}` characters. Alternatively, if you're using an Amazon EKS cluster, you can

delete the "cluster_name" field and value. If you do, the CloudWatch agent detects the cluster name from the Amazon EC2 tags.

3. (Optional) Make further changes to the ConfigMap based on your monitoring requirements, as follows:
 - **metrics_collection_interval** – In the `kubernetes` section, you can specify how often the agent collects metrics. The default is 60 seconds. The default cadvisor collection interval in kubelet is 15 seconds, so don't set this value to less than 15 seconds.
 - **endpoint_override** – In the `logs` section, you can specify the CloudWatch Logs endpoint if you want to override the default endpoint. You might want to do this if you're publishing from a cluster in a VPC and you want the data to go to a VPC endpoint.
 - **force_flush_interval** – In the `logs` section, you can specify the interval for batching log events before they are published to CloudWatch Logs. The default is 5 seconds.
 - **region** – By default, the agent publishes metrics to the Region where the worker node is located. To override this, you can add a `region` field in the agent section: for example, `"region": "us-west-2"`.
 - **statsd** section – If you want the CloudWatch Logs agent to also run as a StatsD listener in each worker node of your cluster, you can add a `statsd` section to the `metrics` section, as in the following example. For information about other StatsD options for this section, see [Retrieve Custom Metrics with StatsD](#) (p. 302).

```
"metrics": {
  "metrics_collected": {
    "statsd": {
      "service_address": ":8125"
    }
  }
}
```

A full example of the JSON section is as follows.

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "kubernetes": {
        "cluster_name": "MyCluster",
        "metrics_collection_interval": 60
      }
    },
    "force_flush_interval": 5,
    "endpoint_override": "logs.us-east-1.amazonaws.com"
  },
  "metrics": {
    "metrics_collected": {
      "statsd": {
        "service_address": ":8125"
      }
    }
  }
}
```

4. Create the ConfigMap in the cluster by running the following command.

```
kubectl apply -f cwagent-configmap.yaml
```


Step 4: Deploy the CloudWatch Agent as a DaemonSet

To finish the installation of the CloudWatch agent and begin collecting container metrics, use the following steps.

To deploy the CloudWatch agent as a DaemonSet

- If you do not want to use StatsD on the cluster, enter the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

- If you do want to use StatsD, follow these steps:
 - a. Download the DaemonSet YAML to your `kubectl` client host by running the following command.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

- b. Uncomment the port section in the `cwagent-daemonset.yaml` file as in the following:

```
ports:
  - containerPort: 8125
    hostPort: 8125
    protocol: UDP
```

- c. Deploy the CloudWatch agent in your cluster by running the following command.

```
kubectl apply -f cwagent-daemonset.yaml
```

2. Validate that the agent is deployed by running the following command.

```
kubectl get pods -n amazon-cloudwatch
```

When complete, the CloudWatch agent creates a log group named `/aws/containerinsights/Cluster_Name/performance` and sends the performance log events to this log group. If you also set up the agent as a StatsD listener, the agent also listens for StatsD metrics on port 8125 with the IP address of the node where the application pod is scheduled.

Troubleshooting

If the agent doesn't deploy correctly, try the following:

- Run the following command to get the list of pods.

```
kubectl get pods -n amazon-cloudwatch
```

- Run the following command and check the events at the bottom of the output.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- Run the following command to check the logs.

```
kubectl logs pod-name -n amazon-cloudwatch
```

Set Up FluentD as a DaemonSet to Send Logs to CloudWatch Logs

To set up FluentD to collect logs from your containers, you can follow the steps in [Quick Start Setup for Container Insights on Amazon EKS \(p. 166\)](#) or you can follow the steps in this section. In the following steps, you set up FluentD as a DaemonSet to send logs to CloudWatch Logs. When you complete this step, FluentD creates the following log groups if they don't already exist.

Log Group Name	Log Source
/aws/containerinsights/ <i>Cluster_Name</i> /application	All log files in /var/log/containers
/aws/containerinsights/ <i>Cluster_Name</i> /host	Logs from /var/log/dmesg, /var/log/secure, and /var/log/messages
/aws/containerinsights/ <i>Cluster_Name</i> /dataplane	The logs in /var/log/journal for kubelet.service, kubeproxy.service, and docker.service.

Step 1: Create a Namespace for CloudWatch

Use the following step to create a Kubernetes namespace called `amazon-cloudwatch` for CloudWatch. You can skip this step if you have already created this namespace.

To create a namespace for CloudWatch

- Enter the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

Step 2: Install FluentD

Start this process by downloading FluentD. When you finish these steps, the deployment creates the following resources on the cluster:

- A service account named `fluentd` in the `amazon-cloudwatch` namespace. This service account is used to run the FluentD DaemonSet. For more information, see [Managing Service Accounts](#) in the Kubernetes Reference.
- A cluster role named `fluentd` in the `amazon-cloudwatch` namespace. This cluster role grants `get`, `list`, and `watch` permissions on pod logs to the `fluentd` service account. For more information, see [API Overview](#) in the Kubernetes Reference.
- A ConfigMap named `fluentd-config` in the `amazon-cloudwatch` namespace. This ConfigMap contains the configuration to be used by FluentD. For more information, see [Configure a Pod to Use a ConfigMap](#) in the Kubernetes Tasks documentation.

To install FluentD

- Create a ConfigMap named `cluster-info` with the cluster name and the AWS Region that the logs will be sent to. Run the following command, updating the placeholders with your cluster and Region names.

```
kubectl create configmap cluster-info \
--from-literal=cluster.name=cluster_name \
--from-literal=logs.region=region_name -n amazon-cloudwatch
```

2. Download and deploy the FluentD DaemonSet to the cluster by running the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/fluentd/fluentd.yaml
```

3. Validate the deployment by running the following command. Each node should have one pod named `fluentd-cloudwatch-*`.

```
kubectl get pods -n amazon-cloudwatch
```

Step 3: Verify the FluentD Setup

To verify your FluentD setup, use the following steps.

To verify the FluentD setup for Container Insights

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Logs**. Make sure that you're in the Region where you deployed FluentD to your containers.

In the list of log groups in the Region, you should see the following:

- `/aws/containerinsights/Cluster_Name/application`
- `/aws/containerinsights/Cluster_Name/host`
- `/aws/containerinsights/Cluster_Name/dataplane`

If you see these log groups, the FluentD setup is verified.

Multiline Log Support

On August 19 2019, we added multiline log support for the logs collected by FluentD.

By default, the multiline log entry starter is any character with no white space. This means that all log lines that start with a character that does not have white space are considered as a new multiline log entry.

If your own application logs use a different multiline starter, you can support them by making two changes in the `fluentd.yaml` file.

First, exclude them from the default multiline support by adding the pathnames of your log files to an `exclude_path` field in the `containers` section of `fluentd.yaml`. The following is an example.

```
<source>
  @type tail
  @id in_tail_container_logs
  @label @containers
  path /var/log/containers/*.log
  exclude_path ["full_pathname_of_log_file*", "full_pathname_of_log_file2*"]
```

Next, add a block for your log files to the `fluentd.yaml` file. The example below is used for the CloudWatch agent's log file, which uses a timestamp regular expression as the multiline starter. You can copy this block and add it to `fluentd.yaml`. Change the indicated lines to reflect your application log file name and the multiline starter that you want to use.

```
<source>
  @type tail
  @id in_tail_cwagent_logs
  @label @cwagentlogs
  path /var/log/containers/cloudwatch-agent*
  pos_file /var/log/cloudwatch-agent.log.pos
  tag *
  read_from_head true
<parse>
  @type json
  time_format %Y-%m-%dT%H:%M:%S.%NZ
</parse>
</source>
```

```
<label @cwagentlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_cwagent
  </filter>

  <filter **>
    @type record_transformer
    @id filter_cwagent_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
  </filter>

  <filter **>
    @type concat
    key log
    multiline_start_regexp /^\[d{4}\[-/\]d{1,2}\[-/\]d{1,2}/
    separator ""
    flush_interval 5
    timeout_label @NORMAL
  </filter>

  <match **>
    @type relabel
    @label @NORMAL
  </match>
</label>
```

Reducing the Log Volume From FluentD (Optional)

By default, we send FluentD application logs and Kubernetes metadata to CloudWatch. If you want to reduce the volume of data being sent to CloudWatch, you can stop one or both of these data sources from being sent to CloudWatch.

To stop FluentD application logs, remove the following section from the `fluentd.yaml` file.

```
<source>
  @type tail
```

```
@id in_tail_fluentd_logs
@label @fluentdlogs
path /var/log/containers/fluentd*
pos_file /var/log/fluentd.log.pos
tag *
read_from_head true
<parse>
  @type json
  time_format %Y-%m-%dT%H:%M:%S.%NZ
</parse>
</source>

<label @fluentdlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_fluentd
  </filter>

  <filter **>
    @type record_transformer
    @id filter_fluentd_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
  </filter>

  <match **>
    @type relabel
    @label @NORMAL
  </match>
</label>
```

To remove Kubernetes metadata from being appended to log events that are sent to CloudWatch, add one line to the `record_transformer` section in the `fluentd.yaml` file. In the log source where you want to remove this metadata, add the following line.

```
remove_keys $.kubernetes.pod_id, $.kubernetes.master_url, $.kubernetes.container_image_id,
$.kubernetes.namespace_id
```

For example:

```
<filter **>
  @type record_transformer
  @id filter_containers_stream_transformer
  <record>
    stream_name ${tag_parts[3]}
  </record>
  remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
$.kubernetes.container_image_id, $.kubernetes.namespace_id
</filter>
```

Troubleshooting

If you don't see these log groups and are looking in the correct Region, check the logs for the FluentD DaemonSet pods to look for the error.

Run the following command and make sure that the status is Running.

```
kubectl get pods -n amazon-cloudwatch
```

In the results of the previous command, note the pod name that starts with `fluentd-cloudwatch`. Use this pod name in the following command.

```
kubectl logs pod_name -n amazon-cloudwatch
```

If the logs have errors related to IAM permissions, check the IAM role attached to the cluster nodes. For more information about the permissions required to run an Amazon EKS cluster, see [Amazon EKS IAM Policies, Roles, and Permissions](#) in the *Amazon EKS User Guide*.

If the pod status is `CreateContainerConfigError`, get the exact error by running the following command.

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

(Optional) Set Up Amazon EKS Control Plane Logging

If you're using Amazon EKS, you can optionally enable Amazon EKS control plane logging, to provide audit and diagnostic logs directly from the Amazon EKS control plane to CloudWatch Logs. For more information, see [Amazon EKS Control Plane Logging](#).

(Optional) Enable App Mesh Envoy Access Logs

You can set up Container Insights FluentD to send App Mesh Envoy access logs to CloudWatch Logs. For more information, see [Access logs](#).

To have Envoy access logs sent to CloudWatch Logs

1. Set up FluentD in the cluster. For more information, see [Set Up FluentD as a DaemonSet to Send Logs to CloudWatch Logs \(p. 170\)](#).
2. Configure Envoy access logs for your virtual nodes. Be sure to configure the log path to be `/dev/stdout` in each virtual node. There are three methods for enabling and configuring the logs:
 - a. Use the App Mesh console. For instructions on how to use the App Mesh console to configure the logs, see [Access logs](#).
 - b. To use the App Mesh CLI to enable or update the Envoy access logs, use the `update-virtual-node` command. For more information, see [update-virtual-node](#).
 - c. To use the YAML file, add the bold logging section of the YAML file shown below when you create or update your virtual node.

```
---
apiVersion: appmesh.k8s.aws/v1beta1
kind: VirtualNode
metadata:
  name: metal-v2
  namespace: {{namespace}}
spec:
  meshName: {{appmeshname}}
  listeners:
    - portMapping:
        port: 9080
        protocol: http
  serviceDiscovery:
    dns:
      hostName: metal-v2.{{namespace}}.svc.cluster.local
  logging:
    accessLog:
      file:
        path: "/dev/stdout"
```

When you have finished, the envoy access logs are sent to the `/aws/containerinsights/Cluster_Name/application` log group.

Updating or Deleting Container Insights on Amazon EKS and Kubernetes

Use the steps in these sections to update your CloudWatch agent container image, or to remove Container Insights from an Amazon EKS or Kubernetes cluster.

Topics

- [Updating the CloudWatch Agent Container Image \(p. 175\)](#)
- [Deleting the CloudWatch Agent and FluentD for Container Insights \(p. 176\)](#)

Updating the CloudWatch Agent Container Image

If you need to update your container image to the latest version, use the steps in this section.

To update your container image

1. Apply the latest `cwagent-serviceaccount.yaml` file by entering the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

2. This step is necessary only for customers who upgraded their containerized CloudWatch agent from a version earlier than 1.226589.0, which was released on August 20, 2019.

In the Configmap file `cwagentconfig`, change the keyword `structuredlogs` to `logs`

- a. First, open the existing `cwagentconfig` in edit mode by entering the following command.

```
kubectl edit cm cwagentconfig -n amazon-cloudwatch
```

In the file, if you see the keyword `structuredlogs`, change it to `logs`

- b. Enter `wq` to save the file and exit edit mode.

3. Apply the latest `cwagent-daemonset.yaml` file by entering the following command.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

You can achieve rolling updates of the CloudWatch agent DaemonSet anytime that you change your configuration in `cwagent-configmap.yaml`. To do so, you must make sure the `.spec.template` section in the `cwagent-daemonset.yaml` file has changes. Otherwise, Kubernetes treats the DaemonSet as unchanged. A common practice is to add the hash value of the ConfigMap into `.spec.template.metadata.annotations.configHash`, as in the following example.

```
yq w -i cwagent-daemonset.yaml spec.template.metadata.annotations.configHash $(kubectl get cm/cwagentconfig -n amazon-cloudwatch -o yaml | sha256sum)
```

This adds a hash value into the `cwagent-daemonset.yaml` file, as in the following example.

```
spec:
```

```
selector:
  matchLabels:
    name: cloudwatch-agent
template:
  metadata:
    labels:
      name: cloudwatch-agent
    annotations:
      configHash: 88915de4cf9c3551a8dc74c0137a3e83569d28c71044b0359c2578d2e0461825
```

Then if you run the following command, the new configuration is picked up.

```
kubectl apply -f cwagent-daemonset.yaml
```

For more information about `yq`, see [yq](#).

Deleting the CloudWatch Agent and FluentD for Container Insights

To delete all resources related to the CloudWatch agent and Fluentd, enter the following command. In this command, **Cluster** is the name of your Amazon EKS or Kubernetes cluster, and **Region** is the name of the Region where the logs are published.

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/quickstart/cwagent-fluentd-quickstart.yaml | sed "s/{{cluster_name}}/Cluster_Name;/s/{{region_name}}/Region/" | kubectl delete -f -
```

Viewing Container Insights Metrics

After you have Container Insights set up and it is collecting metrics, you can view those metrics in the CloudWatch console.

For Container Insights metrics to appear on your dashboard, you must complete the Container Insights setup. For more information, see [Setting Up Container Insights \(p. 157\)](#).

To view Container Insights metrics

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Performance Monitoring**.
3. Use the drop-down boxes near the top to select the type of resource to view, as well as the specific resource.

You can set a CloudWatch alarm on any metric that Container Insights collects. For more information, see [Using Amazon CloudWatch Alarms \(p. 72\)](#)

Using CloudWatch Logs Insights to View Container Insights Data

Container Insights collects metrics by using performance log events, which are stored in CloudWatch Logs. You can use CloudWatch Logs Insights queries for additional views of your container data.

For more information about CloudWatch Logs Insights, see [Analyze Log Data with CloudWatch Logs Insights](#). For more information about the log fields you can use in queries, see [Container Insights Performance Log Events for Amazon EKS and Kubernetes \(p. 188\)](#).

To use CloudWatch Logs Insights to query your container metric data

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Insights**.

Near the top of the screen is the query editor. When you first open CloudWatch Logs Insights, this box contains a default query that returns the 20 most recent log events.

3. In the box above the query editor, select one of the Container Insights log groups to query. For the following example queries to work, the log group name must end with **performance**.

When you select a log group, CloudWatch Logs Insights automatically detects fields in the data in the log group and displays them in **Discovered fields** in the right pane. It also displays a bar graph of log events in this log group over time. This bar graph shows the distribution of events in the log group that matches your query and time range, not only the events displayed in the table.

4. In the query editor, replace the default query with the following query and choose **Run query**.

```
STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName
| SORT avg_node_cpu_utilization DESC
```

This query shows a list of nodes, sorted by average node CPU utilization.

5. To try another example, replace that query with another query and choose **Run query**. More sample queries are listed later on this page.

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

This query displays a list of your pods, sorted by average number of container restarts.

6. If you want to try another query, you can use include fields in the list at the right of the screen. For more information about query syntax, see [CloudWatch Logs Insights Query Syntax](#).

To see lists of your resources

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Resources**.
3. The default view is a list of your resources being monitored by Container Insights, and alarms that you have set on these resources. To see a visual map of the resources, choose **Map view**.
4. From the map view, you can pause your pointer over any resource in the map to see basic metrics about that resource. You can choose any resource to see more detailed graphs about the resource.

Other Sample Queries for Container Insights

List of your pods, sorted by average number of container restarts

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

Pods requested vs. pods running

```
fields @timestamp, @message
| sort @timestamp desc
| filter Type="Pod"
```

```
| stats min(pod_number_of_containers) as requested, min(pod_number_of_running_containers)
  as running, ceil(avg(pod_number_of_containers-pod_number_of_running_containers)) as
  pods_missing by kubernetes.pod_name
| sort pods_missing desc
```

Count of cluster node failures

```
stats avg(cluster_failed_node_count) as CountOfNodeFailures
| filter Type="Cluster"
| sort @timestamp desc
```

Application log errors by container name

```
stats count() as countoferrors by kubernetes.container_name
| filter stream="stderr"
| sort countoferrors desc
```

Disk usage by container name

```
stats floor(avg(container_filesystem_usage/1024)) as container_filesystem_usage_avg_kb by
  InstanceId, kubernetes.container_name, device
| filter Type="ContainerFS"
| sort container_filesystem_usage_avg_kb desc
```

CPU usage by container name

```
stats pct(container_cpu_usage_total, 50) as CPUPercMedian by kubernetes.container_name
| filter Type="Container"
```

Metrics Collected by Container Insights

Container Insights collects one set of metrics for Amazon ECS and AWS Fargate, and a different set for Amazon EKS and Kubernetes.

Topics

- [Amazon ECS Container Insights Metrics \(p. 178\)](#)
- [Amazon EKS and Kubernetes Container Insights Metrics \(p. 182\)](#)

Amazon ECS Container Insights Metrics

The following table lists the metrics and dimensions that Container Insights collects for Amazon ECS. These metrics are in the `ECS/ContainerInsights` namespace. For more information, see [Metrics \(p. 3\)](#).

If you do not see any Container Insights metrics in your console, be sure that you have completed the setup of Container Insights. Metrics do not appear before Container Insights has been set up completely. For more information, see [Setting Up Container Insights \(p. 157\)](#).

Note

The network and disk metrics collected are cumulative per Amazon ECS task. Because these cumulative metrics are aggregated in CloudWatch and then a `RATE` function is applied, it is expected to see spikes in those metrics. For example, if one out of two Amazon ECS tasks in a cluster stops running, the `RATE` on the cumulative network metric will show a negative spike for that data point equal to the network metric data collected from the Amazon ECS task that has just stopped. The CloudWatch team is aware of this behavior and is researching how best to collect, monitor, and alarm on cumulative metrics.

For more information about the `RATE` function, see [Metric Math Syntax and Functions \(p. 53\)](#).

The following metrics are available when you complete the steps in [Setting Up Container Insights on Amazon ECS for Cluster- and Service-Level Metrics \(p. 157\)](#)

Metric Name	Dimensions	Description
ContainerInstanceCount	ClusterName	The number of EC2 instances running the Amazon ECS agent that are registered with a cluster.
CpuUtilized	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The CPU units used by tasks in the resource that is specified by the dimension set that you're using. This metric is collected only for tasks that have a defined CPU reservation in their task definition.
CpuReserved	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The CPU units reserved by tasks in the resource that is specified by the dimension set that you're using. This metric is collected only for tasks that have a defined CPU reservation in their task definition.
DeploymentCount	ServiceName, ClusterName	The number of deployments in an Amazon ECS service.
DesiredTaskCount	ClusterName	The desired number of tasks for an Amazon ECS service.
MemoryUtilized	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The memory being used by tasks in the resource that is specified by the dimension set that you're using. This metric is collected only for tasks that have a defined memory reservation in their task definition.
MemoryReserved	TaskDefinitionFamily, ClusterName	The memory that is reserved by tasks

Metric Name	Dimensions	Description
	ServiceName, ClusterName ClusterName	in the resource that is specified by the dimension set that you're using. This metric is collected only for tasks that have a defined memory reservation in their task definition.
NetworkRxBytes	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The number of bytes received by the resource that is specified by the dimensions that you're using. This metric is available only for containers in bridge network mode. It is not available for containers in awsvpc network mode or host network mode.
NetworkTxBytes	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The number of bytes transmitted by the resource that is specified by the dimensions that you're using. This metric is available only for containers in bridge network mode. It is not available for containers in awsvpc network mode or host network mode.
PendingTaskCount	ServiceName, ClusterName	The number of tasks currently in the PENDING state.
RunningTaskCount	ServiceName, ClusterName	The number of tasks currently in the RUNNING state.
ServiceCount	ServiceName	The number of services in the cluster.
StorageReadBytes	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The number of bytes read from storage in the resource that is specified by the dimensions that you're using.

Metric Name	Dimensions	Description
StorageWriteBytes	TaskDefinitionFamily, ClusterName ServiceName, ClusterName ClusterName	The number of bytes written to storage in the resource that is specified by the dimensions that you're using.
TaskCount	ServiceName	The number of tasks running in the service.
TaskSetCount	ServiceName, ClusterName	The number of task sets in the service.

The following metrics are available when you complete the steps in [Deploying the CloudWatch Agent to Collect EC2 Instance-Level Metrics on Amazon ECS \(p. 159\)](#)

Metric Name	Dimensions	Description
instance_cpu_limit	ClusterName	The maximum number of CPU units that can be assigned to a single EC2 Instance in the cluster.
instance_cpu_reserved_capacity	ClusterName EC2InstanceId, ContainerInstanceId,ClusterName	The percentage of CPU currently being reserved on a single EC2 instance in the cluster.
instance_cpu_usage_total	ClusterName	The number of CPU units being used on a Single EC2 instance in the cluster.
instance_cpu_utilization	ClusterName EC2InstanceId, ContainerInstanceId,ClusterName	The total percentage of CPU units being used on a single EC2 instance in the cluster.
instance_filesystem_utilization	ClusterName EC2InstanceId, ContainerInstanceId,ClusterName	The total percentage of file system capacity being used on a single EC2 instance in the cluster.
instance_memory_limit	ClusterName	The maximum amount of memory, in bytes, that can be assigned to a single EC2 Instance in this cluster.
instance_memory_reserved_capacity	ClusterName EC2InstanceId, ContainerInstanceId,ClusterName	The percentage of Memory currently being reserved on a single

Metric Name	Dimensions	Description
		EC2 Instance in the cluster.
instance_memory_utilization	ClusterName EC2InstanceId, ContainerInstanceId, ClusterName	The total percentage of memory being used on a single EC2 Instance in the cluster.
instance_memory_working_set_bytes	ClusterName	The amount of memory, in bytes, being used on a single EC2 Instance in the cluster.
instance_network_total_bytes	ClusterName	The total number of bytes per second transmitted and received over the network on a single EC2 Instance in the cluster.
instance_number_of_running_tasks	ClusterName	The number of running tasks on a single EC2 Instance in the cluster.

Amazon EKS and Kubernetes Container Insights Metrics

The following table lists the metrics and dimensions that Container Insights collects for Amazon EKS and Kubernetes. These metrics are in the `ContainerInsights` namespace. For more information, see [Metrics \(p. 3\)](#).

If you do not see any Container Insights metrics in your console, be sure that you have completed the setup of Container Insights. Metrics do not appear before Container Insights has been set up completely. For more information, see [Setting Up Container Insights \(p. 157\)](#).

Metric Name	Dimensions	Description
cluster_failed_node_count	ClusterName	The number of failed worker nodes in the cluster.
cluster_node_count	ClusterName	The total number of worker nodes in the cluster.
namespace_number_of_pods	Namespace, ClusterName	The number of pods running per namespace in the resource that is specified by the dimensions that you're using.
node_cpu_limit	ClusterName	The maximum number of CPU units that can

Metric Name	Dimensions	Description
		be assigned to a single node in this cluster.
node_cpu_reserved_capacity	NodeName, ClusterName, InstanceId ClusterName	The percentage of CPU units that are reserved for node components, such as kubelet, kube-proxy, and Docker.
node_cpu_usage_total	ClusterName	The number of CPU units being used on the nodes in the cluster.
node_cpu_utilization	NodeName, ClusterName, InstanceId ClusterName	The total percentage of CPU units being used on the nodes in the cluster.
node_filesystem_utilization	NodeName, ClusterName, InstanceId ClusterName	The total percentage of file system capacity being used on nodes in the cluster.
node_memory_limit	ClusterName	The maximum amount of memory, in bytes, that can be assigned to a single node in this cluster.
node_memory_reserved	NodeName, ClusterName, InstanceId ClusterName	The percentage of memory currently being used on the nodes in the cluster.
node_memory_utilization	NodeName, ClusterName, InstanceId ClusterName	The percentage of memory currently being used by the node or nodes. node_memory_utilization is calculated by $\frac{\text{node_memory_working_set}}{\text{node_memory_limit}}$. It is the percentage of node memory usage over the node memory limitation.
node_memory_working_set	ClusterName	The amount of memory, in bytes, being used in the working set of the nodes in the cluster.

Metric Name	Dimensions	Description
node_network_total_bytes	NodeName, ClusterName, InstanceId ClusterName	The total number of bytes per second transmitted and received over the network per node in a cluster.
node_number_of_running_containers	NodeName, ClusterName, InstanceId ClusterName	The number of running containers per node in a cluster.
node_number_of_running_pods	NodeName, ClusterName, InstanceId ClusterName	The number of running pods per node in a cluster.
pod_cpu_reserved_capacity	PodName, Namespace, ClusterName ClusterName	The CPU capacity that is reserved per pod in a cluster.
pod_cpu_utilization	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	The percentage of CPU units being used by pods.
pod_cpu_utilization_over_limit	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	The percentage of CPU units being used by pods that is over the pod limit.
pod_memory_reserved	PodName, Namespace, ClusterName ClusterName	The percentage of memory that is reserved for pods.
pod_memory_utilization	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	The percentage of memory currently being used by the pod or pods.
pod_memory_utilization_over_limit	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	The percentage of memory that is being used by pods that is over the pod limit.
pod_number_of_container_restarts	PodName, Namespace, ClusterName	The total number of container restarts in a pod.

Metric Name	Dimensions	Description
pod_network_rx_bytes	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	The number of bytes per second being received over the network by the pod.
pod_network_tx_bytes	PodName, Namespace, ClusterName Namespace, ClusterName Service, Namespace, ClusterName ClusterName	The number of bytes per second being transmitted over the network by the pod.
service_number_of_running_pods	Service, Namespace, ClusterName ClusterName	The number of pods running the service or services in the cluster.

Container Insights Performance Log Reference

This section includes reference information about how Container Insights uses performance log events to collect metrics.

Topics

- [Container Insights Performance Log Events for Amazon ECS \(p. 185\)](#)
- [Container Insights Performance Log Events for Amazon EKS and Kubernetes \(p. 188\)](#)
- [Relevant Fields in Performance Log Events for Amazon EKS and Kubernetes \(p. 199\)](#)

Container Insights Performance Log Events for Amazon ECS

The following are examples of the performance log events that Container Insights collects from Amazon ECS.

Type: Container

```
{
  "Version": "0",
  "Type": "Container",
  "ContainerName": "mysql",
  "TaskId": "7c7bce41-8f2f-4673-b1a3-0e57ef51168f",
  "TaskDefinitionFamily": "hello_world",
  "TaskDefinitionRevision": "2",
  "ContainerInstanceId": "12345678-db61-4b1b-b0ec-93ad21467cc9",
  "EC2InstanceId": "i-1234567890123456",
  "ServiceName": "myCIService",
  "ClusterName": "myCICluster",
  "Timestamp": 1561586749104,
  "CpuUtilized": 1.8381139895790504,
  "CpuReserved": 30.0,
  "MemoryUtilized": 295,
  "MemoryReserved": 400,
```

```
"StorageReadBytes": 10817536,  
"StorageWriteBytes": 2085376000,  
"NetworkRxBytes": 6522,  
"NetworkRxDropped": 0,  
"NetworkRxErrors": 0,  
"NetworkRxPackets": 83,  
"NetworkTxBytes": 3904,  
"NetworkTxDropped": 0,  
"NetworkTxErrors": 0,  
"NetworkTxPackets": 44  
}
```

Type: Task

```
{  
  "Version": "0",  
  "Type": "Task",  
  "TaskId": "8b03d710-a0de-4458-b6ee-e9829e9d1440",  
  "TaskDefinitionFamily": "hello_world",  
  "TaskDefinitionRevision": "2",  
  "ContainerInstanceId": "12345678-e797-496b-9d70-4edfda2700a4",  
  "EC2InstanceId": "i-1234567890123456",  
  "ServiceName": "myCIService",  
  "ClusterName": "myCICluster",  
  "Timestamp": 1561586269429,  
  "CpuUtilized": 2.4353688568190526,  
  "CpuReserved": 60.0,  
  "MemoryUtilized": 314,  
  "MemoryReserved": 800,  
  "StorageReadBytes": 11124736,  
  "StorageWriteBytes": 2100379648,  
  "NetworkRxBytes": 12062,  
  "NetworkRxDropped": 0,  
  "NetworkRxErrors": 0,  
  "NetworkRxPackets": 149,  
  "NetworkTxBytes": 8064,  
  "NetworkTxDropped": 0,  
  "NetworkTxErrors": 0,  
  "NetworkTxPackets": 96,  
  "CloudWatchMetrics": [  
    {  
      "Namespace": "ECS/ContainerInsights",  
      "Metrics": [  
        {  
          "Name": "CpuUtilized",  
          "Unit": "None"  
        },  
        {  
          "Name": "CpuReserved",  
          "Unit": "None"  
        },  
        {  
          "Name": "MemoryUtilized",  
          "Unit": "Megabytes"  
        },  
        {  
          "Name": "MemoryReserved",  
          "Unit": "Megabytes"  
        },  
        {  
          "Name": "StorageReadBytes",  
          "Unit": "Bytes/Second"  
        },  
        {  
          "Name": "StorageWriteBytes",
```

```
        "Unit": "Bytes/Second"
      },
      {
        "Name": "NetworkRxBytes",
        "Unit": "Bytes/Second"
      },
      {
        "Name": "NetworkTxBytes",
        "Unit": "Bytes/Second"
      }
    ],
    "Dimensions": [
      [
        "ClusterName"
      ],
      [
        "ServiceName",
        "ClusterName"
      ],
      [
        "ClusterName",
        "TaskDefinitionFamily"
      ]
    ]
  }
}
```

Type: Service

```
{
  "Version": "0",
  "Type": "Service",
  "ServiceName": "myCIService",
  "ClusterName": "myCICluster",
  "Timestamp": 1561586460000,
  "DesiredTaskCount": 2,
  "RunningTaskCount": 2,
  "PendingTaskCount": 0,
  "DeploymentCount": 1,
  "TaskSetCount": 0,
  "CloudWatchMetrics": [
    {
      "Namespace": "ECS/ContainerInsights",
      "Metrics": [
        {
          "Name": "DesiredTaskCount",
          "Unit": "Count"
        },
        {
          "Name": "RunningTaskCount",
          "Unit": "Count"
        },
        {
          "Name": "PendingTaskCount",
          "Unit": "Count"
        },
        {
          "Name": "DeploymentCount",
          "Unit": "Count"
        },
        {
          "Name": "TaskSetCount",
          "Unit": "Count"
        }
      ]
    }
  ]
}
```

```
    ],
    "Dimensions": [
      [
        "ServiceName",
        "ClusterName"
      ]
    ]
  }
]
```

Type: Cluster

```
{
  "Version": "0",
  "Type": "Cluster",
  "ClusterName": "myCICluster",
  "Timestamp": 1561587300000,
  "TaskCount": 5,
  "ContainerInstanceCount": 5,
  "ServiceCount": 2,
  "CloudWatchMetrics": [
    {
      "Namespace": "ECS/ContainerInsights",
      "Metrics": [
        {
          "Name": "TaskCount",
          "Unit": "Count"
        },
        {
          "Name": "ContainerInstanceCount",
          "Unit": "Count"
        },
        {
          "Name": "ServiceCount",
          "Unit": "Count"
        }
      ]
    },
    "Dimensions": [
      [
        "ClusterName"
      ]
    ]
  ]
}
```

Container Insights Performance Log Events for Amazon EKS and Kubernetes

The following are examples of the performance log events that Container Insights collects from Amazon EKS and Kubernetes clusters.

Type: Node

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
```

```
{
  "Unit": "Percent",
  "Name": "node_cpu_utilization"
},
{
  "Unit": "Percent",
  "Name": "node_memory_utilization"
},
{
  "Unit": "Bytes/Second",
  "Name": "node_network_total_bytes"
},
{
  "Unit": "Percent",
  "Name": "node_cpu_reserved_capacity"
},
{
  "Unit": "Percent",
  "Name": "node_memory_reserved_capacity"
},
{
  "Unit": "Count",
  "Name": "node_number_of_running_pods"
},
{
  "Unit": "Count",
  "Name": "node_number_of_running_containers"
}
],
"Dimensions": [
  [
    "NodeName",
    "InstanceId",
    "ClusterName"
  ]
],
"Namespace": "ContainerInsights"
},
{
  "Metrics": [
    {
      "Unit": "Percent",
      "Name": "node_cpu_utilization"
    },
    {
      "Unit": "Percent",
      "Name": "node_memory_utilization"
    },
    {
      "Unit": "Bytes/Second",
      "Name": "node_network_total_bytes"
    },
    {
      "Unit": "Percent",
      "Name": "node_cpu_reserved_capacity"
    },
    {
      "Unit": "Percent",
      "Name": "node_memory_reserved_capacity"
    },
    {
      "Unit": "Count",
      "Name": "node_number_of_running_pods"
    },
    {
      "Unit": "Count",
```

```

        "Name": "node_number_of_running_containers"
    },
    {
        "Name": "node_cpu_usage_total"
    },
    {
        "Name": "node_cpu_limit"
    },
    {
        "Unit": "Bytes",
        "Name": "node_memory_working_set"
    },
    {
        "Unit": "Bytes",
        "Name": "node_memory_limit"
    }
],
"Dimensions": [
    [
        "ClusterName"
    ]
],
"Namespace": "ContainerInsights"
}
],
"ClusterName": "myCICluster",
"InstanceId": "i-1234567890123456",
"InstanceType": "t3.xlarge",
"NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
"Sources": [
    "cadvisor",
    "/proc",
    "pod",
    "calculated"
],
"Timestamp": "1567096682364",
"Type": "Node",
"Version": "0",
"kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
},
"node_cpu_limit": 4000,
"node_cpu_request": 1130,
"node_cpu_reserved_capacity": 28.249999999999996,
"node_cpu_usage_system": 33.794636630852764,
"node_cpu_usage_total": 136.47852169244098,
"node_cpu_usage_user": 71.67075111567326,
"node_cpu_utilization": 3.4119630423110245,
"node_memory_cache": 3103297536,
"node_memory_failcnt": 0,
"node_memory_hierarchical_pgfault": 0,
"node_memory_hierarchical_pgmajfault": 0,
"node_memory_limit": 16624865280,
"node_memory_mapped_file": 406646784,
"node_memory_max_usage": 4230746112,
"node_memory_pgfault": 0,
"node_memory_pgmajfault": 0,
"node_memory_request": 1115684864,
"node_memory_reserved_capacity": 6.7109407818311055,
"node_memory_rss": 798146560,
"node_memory_swap": 0,
"node_memory_usage": 3901444096,
"node_memory_utilization": 6.601302600149552,
"node_memory_working_set": 1097457664,
"node_network_rx_bytes": 35918.392817386324,
"node_network_rx_dropped": 0,

```

```
"node_network_rx_errors": 0,  
"node_network_rx_packets": 157.67565245448117,  
"node_network_total_bytes": 68264.20276554905,  
"node_network_tx_bytes": 32345.80994816272,  
"node_network_tx_dropped": 0,  
"node_network_tx_errors": 0,  
"node_network_tx_packets": 154.21455923431654,  
"node_number_of_running_containers": 16,  
"node_number_of_running_pods": 13  
}
```

Type: NodeFS

```
{  
  "AutoScalingGroupName": "eksctl-myCICluster-nodgroup-standard-workers-  
NodeGroup-1174PV2WHZAYU",  
  "CloudWatchMetrics": [  
    {  
      "Metrics": [  
        {  
          "Unit": "Percent",  
          "Name": "node_filesystem_utilization"  
        }  
      ],  
      "Dimensions": [  
        [  
          "NodeName",  
          "InstanceId",  
          "ClusterName"  
        ],  
        [  
          "ClusterName"  
        ]  
      ],  
      "Namespace": "ContainerInsights"  
    }  
  ],  
  "ClusterName": "myCICluster",  
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",  
  "InstanceId": "i-1234567890123456",  
  "InstanceType": "t3.xlarge",  
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",  
  "Sources": [  
    "cadvisor",  
    "calculated"  
  ],  
  "Timestamp": "1567097939726",  
  "Type": "NodeFS",  
  "Version": "0",  
  "device": "/dev/nvme0n1p1",  
  "fstype": "vfs",  
  "kubernetes": {  
    "host": "ip-192-168-75-26.us-west-2.compute.internal"  
  },  
  "node_filesystem_available": 17298395136,  
  "node_filesystem_capacity": 21462233088,  
  "node_filesystem_inodes": 10484720,  
  "node_filesystem_inodes_free": 10367158,  
  "node_filesystem_usage": 4163837952,  
  "node_filesystem_utilization": 19.400767547940255  
}
```

Type: NodeDiskIO

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeDiskIO",
  "Version": "0",
  "device": "/dev/nvme0n1",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_diskio_io_service_bytes_async": 9750.505814277016,
  "node_diskio_io_service_bytes_read": 0,
  "node_diskio_io_service_bytes_sync": 230.6174506688036,
  "node_diskio_io_service_bytes_total": 9981.123264945818,
  "node_diskio_io_service_bytes_write": 9981.123264945818,
  "node_diskio_io_serviced_async": 1.153087253344018,
  "node_diskio_io_serviced_read": 0,
  "node_diskio_io_serviced_sync": 0.03603397666700056,
  "node_diskio_io_serviced_total": 1.1891212300110185,
  "node_diskio_io_serviced_write": 1.1891212300110185
}
```

Type: NodeNet

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeNet",
  "Version": "0",
  "interface": "eni972f6bfa9a0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_interface_network_rx_bytes": 3163.008420864309,
  "node_interface_network_rx_dropped": 0,
  "node_interface_network_rx_errors": 0,
  "node_interface_network_rx_packets": 16.575629266820258,
  "node_interface_network_total_bytes": 3518.3935157426017,
  "node_interface_network_tx_bytes": 355.385094878293,
  "node_interface_network_tx_dropped": 0,
  "node_interface_network_tx_errors": 0,
  "node_interface_network_tx_packets": 3.9997714100370625
}
```

Type: Pod


```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodgroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_rx_bytes"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_tx_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization_over_pod_limit"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization_over_pod_limit"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_reserved_capacity"
        }
      ],
      "Dimensions": [
        [
```

```
        "PodName",
        "Namespace",
        "ClusterName"
    ],
    [
        "ClusterName"
    ]
],
"Namespace": "ContainerInsights"
},
{
    "Metrics": [
        {
            "Unit": "Count",
            "Name": "pod_number_of_container_restarts"
        }
    ],
    "Dimensions": [
        [
            "PodName",
            "Namespace",
            "ClusterName"
        ]
    ],
    "Namespace": "ContainerInsights"
}
],
"ClusterName": "myCICluster",
"InstanceId": "i-1234567890123456",
"InstanceType": "t3.xlarge",
"Namespace": "amazon-cloudwatch",
"NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
"PodName": "cloudwatch-agent-statsd",
"Service": "cloudwatch-agent-statsd",
"Sources": [
    "cadvisor",
    "pod",
    "calculated"
],
"Timestamp": "1567097351092",
"Type": "Pod",
"Version": "0",
"kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
        "app": "cloudwatch-agent-statsd",
        "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
        {
            "owner_kind": "Deployment",
            "owner_name": "cloudwatch-agent-statsd"
        }
    ]
},
"service_name": "cloudwatch-agent-statsd"
},
"pod_cpu_limit": 200,
"pod_cpu_request": 200,
"pod_cpu_reserved_capacity": 5,
"pod_cpu_usage_system": 1.4504841104992765,
"pod_cpu_usage_total": 5.817016867430125,
"pod_cpu_usage_user": 1.1281543081661038,
"pod_cpu_utilization": 0.14542542168575312,
```

```
"pod_cpu_utilization_over_pod_limit": 2.9085084337150624,
"pod_memory_cache": 8192,
"pod_memory_failcnt": 0,
"pod_memory_hierarchical_pgfault": 0,
"pod_memory_hierarchical_pgmajfault": 0,
"pod_memory_limit": 104857600,
"pod_memory_mapped_file": 0,
"pod_memory_max_usage": 25268224,
"pod_memory_pgfault": 0,
"pod_memory_pgmajfault": 0,
"pod_memory_request": 104857600,
"pod_memory_reserved_capacity": 0.6307275170893897,
"pod_memory_rss": 22777856,
"pod_memory_swap": 0,
"pod_memory_usage": 25141248,
"pod_memory_utilization": 0.10988455961791709,
"pod_memory_utilization_over_pod_limit": 17.421875,
"pod_memory_working_set": 18268160,
"pod_network_rx_bytes": 9880.697124714186,
"pod_network_rx_dropped": 0,
"pod_network_rx_errors": 0,
"pod_network_rx_packets": 107.80005532263283,
"pod_network_total_bytes": 10158.829201483635,
"pod_network_tx_bytes": 278.13207676944796,
"pod_network_tx_dropped": 0,
"pod_network_tx_errors": 0,
"pod_network_tx_packets": 1.146027574644318,
"pod_number_of_container_restarts": 0,
"pod_number_of_containers": 1,
"pod_number_of_running_containers": 1,
"pod_status": "Running"
}
```

Type: PodNet

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodgroup-standard-workers-NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097351092",
  "Type": "PodNet",
  "Version": "0",
  "interface": "eth0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
```

```
        "owner_name": "cloudwatch-agent-statsd"
    },
    ],
    "service_name": "cloudwatch-agent-statsd"
},
"pod_interface_network_rx_bytes": 9880.697124714186,
"pod_interface_network_rx_dropped": 0,
"pod_interface_network_rx_errors": 0,
"pod_interface_network_rx_packets": 107.80005532263283,
"pod_interface_network_total_bytes": 10158.829201483635,
"pod_interface_network_tx_bytes": 278.13207676944796,
"pod_interface_network_tx_dropped": 0,
"pod_interface_network_tx_errors": 0,
"pod_interface_network_tx_packets": 1.146027574644318
}
```

Type: Container

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-sample",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "Container",
  "Version": "0",
  "container_cpu_limit": 200,
  "container_cpu_request": 200,
  "container_cpu_usage_system": 1.87958283771964,
  "container_cpu_usage_total": 6.159993652997942,
  "container_cpu_usage_user": 1.6707403001952357,
  "container_cpu_utilization": 0.15399984132494854,
  "container_memory_cache": 8192,
  "container_memory_failcnt": 0,
  "container_memory_hierarchical_pgfault": 0,
  "container_memory_hierarchical_pgmajfault": 0,
  "container_memory_limit": 104857600,
  "container_memory_mapped_file": 0,
  "container_memory_max_usage": 24580096,
  "container_memory_pgfault": 0,
  "container_memory_pgmajfault": 0,
  "container_memory_request": 104857600,
  "container_memory_rss": 22736896,
  "container_memory_swap": 0,
  "container_memory_usage": 24453120,
  "container_memory_utilization": 0.10574541028701798,
  "container_memory_working_set": 17580032,
  "container_status": "Running",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id": "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
  },
  "host": "ip-192-168-75-26.us-west-2.compute.internal",
  "labels": {
    "app": "cloudwatch-agent-statsd",
  }
}
```

```
    "pod-template-hash": "df44f855f"
  },
  "namespace_name": "amazon-cloudwatch",
  "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
  "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
  "pod_owners": [
    {
      "owner_kind": "Deployment",
      "owner_name": "cloudwatch-agent-statsd"
    }
  ],
  "service_name": "cloudwatch-agent-statsd"
},
"number_of_container_restarts": 0
}
```

Type: ContainerFS

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "ContainerFS",
  "Version": "0",
  "container_filesystem_available": 0,
  "container_filesystem_capacity": 21462233088,
  "container_filesystem_usage": 24576,
  "container_filesystem_utilization": 0.0001145081217748071,
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id": "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  }
}
```

Type: Cluster

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "cluster_node_count"
        },
        {
          "Unit": "Count",
          "Name": "cluster_failed_node_count"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "Cluster",
  "Version": "0",
  "cluster_failed_node_count": 0,
  "cluster_node_count": 3
}
```

Type: ClusterService

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "service_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "apiserver"
  ],
}
```

```
"Timestamp": "1567097534160",
"Type": "ClusterService",
"Version": "0",
"kubernetes": {
  "namespace_name": "amazon-cloudwatch",
  "service_name": "cloudwatch-agent-statsd"
},
"service_number_of_running_pods": 1
}
```

Type: ClusterNamespace

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "namespace_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097594160",
  "Type": "ClusterNamespace",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch"
  },
  "namespace_number_of_running_pods": 7
}
```

Relevant Fields in Performance Log Events for Amazon EKS and Kubernetes

For Amazon EKS and Kubernetes, the containerized CloudWatch agent emits data as performance log events. This enables CloudWatch to ingest and store high-cardinality data. CloudWatch uses the data in the performance log events to create aggregated CloudWatch metrics at the cluster, node, and pod levels without the need to lose granular details.

The following table lists the fields in these performance log events that are relevant to the collection of Container Insights metric data. You can use CloudWatch Logs Insights to query for any of these fields to collect data or investigate issues. For more information, see [Analyze Log Data With CloudWatch Logs Insights](#).

Amazon CloudWatch User Guide
Relevant Fields in Performance Log
Events for Amazon EKS and Kubernetes

Type	Log field	Source	Formula or Notes
Pod	pod_cpu_utilization	Calculated	Formula: $\text{pod_cpu_usage_total} / \text{node_cpu_limit}$
Pod	pod_cpu_usage_total pod_cpu_usage_total is reported in millicores.	cadvisor	
Pod	pod_cpu_limit	Calculated	Formula: $\text{sum}(\text{container_cpu_limit})$ If any containers in the pod don't have a CPU limit defined, this field doesn't appear in the log event.
Pod	pod_cpu_request	Calculated	Formula: $\text{sum}(\text{container_cpu_request})$ container_cpu_request isn't guaranteed to be set. Only the ones that are set are included in the sum.
Pod	pod_cpu_utilization_over_pod_limit	Calculated	Formula: $\text{pod_cpu_usage_total} / \text{pod_cpu_limit}$
Pod	pod_cpu_reserved_capacity	Calculated	Formula: $\text{pod_cpu_request} / \text{node_cpu_limit}$
Pod	pod_memory_utilization	Calculated	Formula: $\text{pod_memory_working_set} / \text{node_memory_limit}$ It is the percentage of pod memory usage over the node memory limitation.
Pod	pod_memory_working_set	cadvisor	
Pod	pod_memory_limit	Calculated	Formula: $\text{sum}(\text{container_memory_limit})$ If any containers in the pod don't

Amazon CloudWatch User Guide
Relevant Fields in Performance Log
Events for Amazon EKS and Kubernetes

Type	Log field	Source	Formula or Notes
			have a memory limit defined, this field doesn't appear in the log event.
Pod	pod_memory_request	Calculated	Formula: $\text{sum}(\text{container_memory_request})$ <p>container_memory_request isn't guaranteed to be set. Only the ones that are set are included in the sum.</p>
Pod	pod_memory_utilization_over_pod_limit	Calculated	Formula: $\frac{\text{pod_memory_working_set}}{\text{pod_memory_limit}}$ <p>If any containers in the pod don't have a memory limit defined, this field doesn't appear in the log event.</p>
Pod	pod_memory_reserved_capacity	Calculated	Formula: $\frac{\text{pod_memory_request}}{\text{node_memory_limit}}$
Pod	pod_network_tx_bytes	Calculated	Formula: $\text{sum}(\text{pod_interface_network_tx_bytes})$ <p>This data is available for all the network interfaces per pod. The CloudWatch agent calculates the total and adds metric extraction rules.</p>
Pod	pod_network_rx_bytes	Calculated	Formula: $\text{sum}(\text{pod_interface_network_rx_bytes})$
Pod	pod_network_total_bytes	Calculated	Formula: $\text{pod_network_rx_bytes} + \text{pod_network_tx_bytes}$

Amazon CloudWatch User Guide
Relevant Fields in Performance Log
Events for Amazon EKS and Kubernetes

Type	Log field	Source	Formula or Notes
PodNet	pod_interface_network_rx_bytes	cadvisor	This data is network rx bytes per second of a pod network interface.
PodNet	pod_interface_network_tx_bytes	cadvisor	This data is network tx bytes per second of a pod network interface.
Container	container_cpu_usage_total	cadvisor	
Container	container_cpu_limit	cadvisor	Not guaranteed to be set. It's not emitted if it's not set.
Container	container_cpu_request	cadvisor	Not guaranteed to be set. It's not emitted if it's not set.
Container	container_memory_working_set	cadvisor	
Container	container_memory_limit	pod	Not guaranteed to be set. It's not emitted if it's not set.
Container	container_memory_request	pod	Not guaranteed to be set. It's not emitted if it's not set.
ContainerFS	container_filesystem_capacity	pod	This data is available per disk device.
ContainerFS	container_filesystem_usage	pod	This data is available per disk device.
ContainerFS	container_filesystem_utilization	Calculated	Formula: $\frac{\text{container_filesystem_usage}}{\text{container_filesystem_capacity}}$ <p>This data is available per device name.</p>
Node	node_cpu_utilization	Calculated	Formula: $\frac{\text{node_cpu_usage_total}}{\text{node_cpu_limit}}$

Amazon CloudWatch User Guide
Relevant Fields in Performance Log
Events for Amazon EKS and Kubernetes

Type	Log field	Source	Formula or Notes
Node	node_cpu_usage_total	cadvisor	
Node	node_cpu_limit	/proc	
Node	node_cpu_request	Calculated	Formula: sum(pod_cpu_request)
Node	node_cpu_reserved_capacity	Calculated	Formula: node_cpu_request / node_cpu_limit
Node	node_memory_utilization	Calculated	Formula: node_memory_working_set node_memory_limit
Node	node_memory_working_set	cadvisor	
Node	node_memory_limit	/proc	
Node	node_memory_request	Calculated	Formula: sum(pod_memory_request)
Node	node_memory_reserved_capacity	Calculated	Formula: node_memory_request / node_memory_limit
Node	node_network_rx_bytes	Calculated	Formula: sum(node_interface_network_rx_bytes)
Node	node_network_tx_bytes	Calculated	Formula: sum(node_interface_network_tx_bytes)
Node	node_network_total_bytes	Calculated	Formula: node_network_rx_bytes + node_network_tx_bytes
Node	node_number_of_running_pods	Pod List	
Node	node_number_of_running_containers	Pod List	
NodeNet	node_interface_network_rx_bytes	cadvisor	This data is network rx bytes per second of a worker node network interface.
NodeNet	node_interface_network_tx_bytes	cadvisor	This data is network tx bytes per second of a worker node network interface.
NodeFS	node_filesystem_capacity	cadvisor	

Type	Log field	Source	Formula or Notes
NodeFS	node_filesystem_usage	cadvisor	
NodeFS	node_filesystem_utilization	Calculated	Formula: $\frac{\text{node_filesystem_usage}}{\text{node_filesystem_capacity}}$ <p>This data is available per device name.</p>
Cluster	cluster_failed_node_count	API Server	
Cluster	cluster_node_count	API Server	
Service	service_number_of_running_pods	API Server	
Namespace	namespace_number_of_running_pods	API Server	

Metrics Calculation Examples

This section includes examples that show how some of the values in the preceding table are calculated.

Suppose that you have a cluster in the following state.

```

Node1
  node_cpu_limit = 4
  node_cpu_usage_total = 3

  Pod1
    pod_cpu_usage_total = 2

    Container1
      container_cpu_limit = 1
      container_cpu_request = 1
      container_cpu_usage_total = 0.8

    Container2
      container_cpu_limit = null
      container_cpu_request = null
      container_cpu_usage_total = 1.2

  Pod2
    pod_cpu_usage_total = 0.4

    Container3
      container_cpu_limit = 1
      container_cpu_request = 0.5
      container_cpu_usage_total = 0.4

Node2
  node_cpu_limit = 8
  node_cpu_usage_total = 1.5

  Pod3
    pod_cpu_usage_total = 1

    Container4
      container_cpu_limit = 2

```

```
container_cpu_request = 2  
container_cpu_usage_total = 1
```

The following table shows how pod CPU metrics are calculated using this data.

Metric	Formula	Pod1	Pod2	Pod3
pod_cpu_utilization	$\text{pod_cpu_usage_total} / \text{node_cpu_limit}$	$2 / 4 = 50\%$	$0.4 / 4 = 10\%$	$1 / 8 = 12.5\%$
pod_cpu_utilization_over_pod_limit	$\text{pod_cpu_usage_total} / \text{sum}(\text{container_cpu_limit})$	N/A because CPU limit for Container2 isn't defined	$0.4 / 1 = 40\%$	$1 / 2 = 50\%$
pod_cpu_reserved_capacity	$\text{sum}(\text{container_cpu_request}) / \text{node_cpu_limit}$	$(1 + 0) / 4 = 25\%$	$0.5 / 4 = 12.5\%$	$2 / 8 = 25\%$

The following table shows how node CPU metrics are calculated using this data.

Metric	Formula	Node1	Node2
node_cpu_utilization	$\text{node_cpu_usage_total} / \text{node_cpu_limit}$	$3 / 4 = 75\%$	$1.5 / 8 = 18.75\%$
node_cpu_reserved_capacity	$\text{sum}(\text{pod_cpu_request}) / \text{node_cpu_limit}$	$1.5 / 4 = 37.5\%$	$2 / 8 = 25\%$

Container Insights Prometheus Metrics Monitoring

Support for Prometheus metrics is in beta. The beta is open to all AWS accounts and you do not need to request access. Features may be added or changed before announcing General Availability. Don't hesitate to contact us with any feedback or let us know if you would like to be informed when updates are made by emailing us at containerinsightsfeedback@amazon.com

CloudWatch Container Insights monitoring for Prometheus automates the discovery of Prometheus metrics from containerized systems and workloads. Prometheus is an open-source systems monitoring and alerting toolkit. For more information, see [What is Prometheus?](#) in the Prometheus documentation.

For this beta, discovering Prometheus metrics is supported for [Amazon Elastic Kubernetes Service](#) and [Kubernetes](#) clusters running on Amazon EC2 instances. In this beta release, only the Prometheus counter and gauge metric types are collected. Support for histogram and summary metrics is planned for an upcoming release.

The Container Insights Prometheus solution automatically collects metrics from the following containerized workloads and systems, and you can configure it to collect Prometheus metrics from other containerized services and applications.

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy

You can adopt Prometheus as an open-source and open-standard method to ingest custom metrics in CloudWatch. The CloudWatch agent with Prometheus support discovers and collects Prometheus metrics to monitor, troubleshoot, and alarm on application performance degradation and failures faster. This also reduces the number of monitoring tools required to improve observability.

Container Insights Prometheus support involves pay-per-use of metrics and logs, including collecting, storing, and analyzing. For more information, see [Amazon CloudWatch Pricing](#).

Install the CloudWatch Agent with Prometheus Metrics Collection

Support for Prometheus metrics is in beta. The beta is open to all AWS accounts and you do not need to request access. Features may be added or changed before announcing General Availability. Don't hesitate to contact us with any feedback or let us know if you would like to be informed when updates are made by emailing us at containerinsightsfeedback@amazon.com

Before following these steps to install the CloudWatch agent for Prometheus metric collection, you must have a cluster running on Amazon EKS or a Kubernetes cluster running on an Amazon EC2 instance.

Topics

- [Setting Up IAM Roles \(p. 206\)](#)
- [Installing the CloudWatch Agent to Collect Prometheus Metrics \(p. 207\)](#)

Setting Up IAM Roles

The first step is to set up the necessary IAM policy in the cluster. You do this by adding the policy to the IAM role used for the cluster.

To set up the IAM policy in a cluster for Prometheus support

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances**.
3. You need to find the prefix of the IAM role name for the cluster. To do this, select the check box next to the name of an instance that is in the cluster, and choose **Actions, Instance Settings, Attach/Replace IAM Role**. Then copy the prefix of the IAM role, such as `eksctl-dev303-workshop-nodegroup`.
4. Open the IAM console at <https://console.aws.amazon.com/iam/>.
5. In the navigation pane, choose **Roles**.
6. Use the search box to find the prefix that you copied earlier in this procedure, and choose that role.
7. Choose **Attach policies**.
8. Use the search box to find **CloudWatchAgentServerPolicy**. Select the check box next to **CloudWatchAgentServerPolicy**, and choose **Attach policy**.

Installing the CloudWatch Agent to Collect Prometheus Metrics

You must install the CloudWatch agent in the cluster to collect the metrics. How to install the agent differs for Amazon EKS clusters and Kubernetes clusters.

Delete Previous Versions of the CloudWatch Agent with Prometheus Support

If you have already installed a version of the CloudWatch agent with Prometheus support in your cluster, you must delete that version by entering the following command. This is necessary only for previous versions of the agent with Prometheus support. You do not need to delete the CloudWatch agent that enables Container Insights without Prometheus support.

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

Installing the CloudWatch Agent on Amazon EKS

To install the CloudWatch agent with Prometheus support on an Amazon EKS cluster, follow these steps.

To install the CloudWatch agent with Prometheus support on an Amazon EKS cluster

1. Enter the following command to check whether the `amazon-cloudwatch` namespace has already been created:

```
kubectl get namespace
```

2. If `amazon-cloudwatch` is not displayed in the results, create it by entering the following command:

```
kubectl create namespace amazon-cloudwatch
```

3. To deploy the agent with the default configuration and have it send data to the AWS Region that it is installed in, enter the following command:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

To have the agent send data to a different Region instead, follow these steps:

- a. Download the YAML file for the agent by entering the following command:

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

- b. Open the file with a text editor, and search for the `cwagentconfig.json` block of the file.
- c. Add the highlighted lines, specifying the Region that you want:

```
cwagentconfig.json: |  
  {  
    "agent": {  
      "region": "us-east-2"  
    },  
    "logs": { ...
```

- d. Save the file and deploy the agent using your updated file.

```
kubectl apply -f prometheus-eks.yaml
```

Installing the CloudWatch Agent on a Kubernetes Cluster

To install the CloudWatch agent with Prometheus support on a cluster running Kubernetes, enter the following command:

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/  
prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-  
prometheus/prometheus-k8s.yaml |  
sed "s/{{cluster_name}}/MyCluster/s/{{region_name}}/region/" |  
kubect1 apply -f -
```

Replace **MyCluster** with the name of the cluster. This name is used in the log group name that stores the log events collected by the agent, and is also used as a dimension for the metrics collected by the agent.

Replace **region** with the name of the AWS Region where you want the metrics to be sent. For example, **us-west-1**.

Verify that the Agent is Running

On both Amazon EKS and Kubernetes clusters, you can enter the following command to confirm that the agent is running.

```
kubect1 get pod -l "app=cwagent-prometheus" -n amazon-cloudwatch
```

If the results include a single CloudWatch agent pod in the Running state, the agent is running and collecting Prometheus metrics. By default the CloudWatch agent collects metrics for App Mesh, NGINX, Memcached, Java/JMX, and HAProxy every minute. For more information about those metrics, see [Prometheus Metrics Collected by the CloudWatch Agent \(p. 224\)](#). For instructions on how to see your Prometheus metrics in CloudWatch, see [Viewing Your Prometheus Metrics \(p. 229\)](#).

You can also configure the CloudWatch agent to collect metrics from other Prometheus exporters. For more information, see [Configuring the CloudWatch Agent for Prometheus Monitoring \(p. 208\)](#).

Configuring the CloudWatch Agent for Prometheus Monitoring

The CloudWatch agent uses a YAML file to configure how it collects Prometheus metrics. This file is called `prometheus-eks.yaml` for Amazon EKS clusters and `prometheus-k8s.yaml` for Kubernetes clusters on Amazon EC2 instances.

This YAML file contains two config map sections:

- The name: `prometheus-config` section contains the settings for Prometheus scraping. It follows the standard Prometheus scrape config as documented in [<scrape_config>](#) in the Prometheus documentation. You use this section to add scrape targets.
- The name: `prometheus-cwagentconfig` section contains the configuration for the CloudWatch agent. You can use this section to configure how the Prometheus metrics are collected by CloudWatch. For example, you whitelist which metrics are to be imported into CloudWatch, and define their dimensions.

To scrape additional Prometheus metrics sources and import those metrics to CloudWatch, you can modify both sections of the YAML file and re-deploy the agent with the updated configuration.

Prometheus Scrape Configuration

The `prometheus-config` config map section in the CloudWatch agent YAML file supports standard Prometheus scrape configurations as documented in [<scrape_config>](#) in the Prometheus documentation. You can edit this section to update the configurations that are already in this file, and add additional Prometheus scraping targets.

By default, the `prometheus-config` section of the `prometheus-eks.yaml` and `prometheus-k8s.yaml` files contains the following global configuration lines:

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
```

- **scrape_interval**— Defines how frequently to scrape targets.
- **scrape_timeout**— Defines how long to wait before a scrape request times out.

You can also define different values for these settings at the job level, to override the global configurations.

Prometheus Scraping Jobs

The CloudWatch agent YAML files already have some default scraping jobs configured. These are defined in the YAML files with `job_name` lines in the `scrape_configs` section. For example, the following default `kubernetes-pod-jmx` section scrapes JMX exporter metrics.

```
- job_name: 'kubernetes-pod-jmx'
  sample_limit: 10000
  metrics_path: /metrics
  kubernetes_sd_configs:
    - role: pod
  relabel_configs:
    - source_labels: [__address__]
      action: keep
      regex: '.*:9404$'
    - action: labelmap
      regex: __meta_kubernetes_pod_label_(.+)
    - action: replace
      source_labels:
        - __meta_kubernetes_namespace
      target_label: Namespace
    - source_labels: [__meta_kubernetes_pod_name]
      action: replace
      target_label: pod_name
    - action: replace
      source_labels:
        - __meta_kubernetes_pod_container_name
      target_label: container_name
    - action: replace
      source_labels:
        - __meta_kubernetes_pod_controller_name
      target_label: pod_controller_name
    - action: replace
      source_labels:
        - __meta_kubernetes_pod_controller_kind
      target_label: pod_controller_kind
    - action: replace
      source_labels:
        - __meta_kubernetes_pod_phase
```

```
target_label: pod_phase
```

Each of these default targets are scraped, and the metrics are sent to CloudWatch in log events using embedded metric format. For more information, see [Ingesting High-Cardinality Logs and Generating Metrics with CloudWatch Embedded Metric Format](#) (p. 387).

These log events are stored in the `/aws/containerinsights/cluster_name/prometheus` log group in CloudWatch Logs

Each scraping job is contained in a different log stream in this log group. For example, the Prometheus scraping job `kubernetes-pod-appmesh-envoy` is defined for App Mesh. All App Mesh Prometheus metrics are sent to the log stream named `/aws/containerinsights/cluster_name>prometheus/kubernetes-pod-appmesh-envoy/`.

To add a new scraping target, you add a new `job_name` section to the `scrape_configs` section of the YAML file, and restart the agent. For an example of this process, see [Tutorial for Adding a New Prometheus Scrape Target: Prometheus KPI Server Metrics](#) (p. 211).

CloudWatch Agent Configuration for Prometheus

The CloudWatch agent YAML file includes a `cwagentconfig.json` section. This section is under `metrics_collected`, which tells the agent how to collect metrics embedded in logs. It includes the following fields:

- **prometheus_config_path**— specifies the Prometheus scrape configuration file path. Do not change this field.
- **metric_declaration**— are sections that specify the array of logs with embedded metric format to be generated. There are `metric_declaration` sections for each Prometheus source that the CloudWatch agent imports from by default. These sections each include the following fields:
 - `label_matcher` is a regular expression that checks the value of the Kubernetes labels listed in `source_labels`. The metrics that match are whitelisted for inclusion in the embedded metric format sent to CloudWatch.
 - `source_labels` specifies the value of the labels that are checked by the `label_matcher` line.
 - `label_separator` specifies the separator to be used in the `label_matcher` line if multiple `source_labels` are specified. The default is `;`. You can see this default used in the `label_matcher` line in the following example.
 - `metric_selectors` is a regular expression that specifies the metrics to be collected and sent to CloudWatch.
 - `dimensions` is the list of labels to be used as CloudWatch dimensions for each selected metric.

See the following `metric_declaration` example.

```
"metric_declaration": [  
  {  
    "source_labels": [ "Service", "Namespace"],  
    "label_matcher": "(.*node-exporter.*|.*kube-dns.*);kube-system$",  
    "dimensions": [  
      [ "Service", "Namespace"]  
    ],  
    "metric_selectors": [  
      "^coredns_dns_request_type_count_total$"  
    ]  
  }  
]
```

This example configures an embedded metric format section to be sent as a log event if the following conditions are met:

- The value of `Service` contains either `node-exporter` or `kube-dns`.
- The value of `Namespace` is `kube-system`.
- The Prometheus metric `coredns_dns_request_type_count_total` contains both `Service` and `Namespace` labels.

The log event that is sent includes the following highlighted section:

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Name": "coredns_dns_request_type_count_total"
        }
      ],
      "Dimensions": [
        [
          "Namespace",
          "Service"
        ]
      ],
      "Namespace": "ContainerInsights/Prometheus"
    }
  ],
  "Namespace": "kube-system",
  "Service": "kube-dns",
  "coredns_dns_request_type_count_total": 2562,
  "eks_amazonaws_com_component": "kube-dns",
  "instance": "192.168.61.254:9153",
  "job": "kubernetes-service-endpoints",
  ...
}
```

Tutorial for Adding a New Prometheus Scrape Target: Prometheus KPI Server Metrics

The Kubernetes API Server exposes Prometheus metrics on endpoints by default. The official example for the Kubernetes API Server scraping configuration is available on [Github](#).

The following tutorial shows how to do the following steps to begin importing Kubernetes API Server metrics into CloudWatch:

- Adding the Prometheus scraping configuration for Kubernetes API Server to the CloudWatch agent YAML file.
- Configuring the embedded metric format metrics definitions in the CloudWatch agent YAML file.
- (Optional) Creating a CloudWatch dashboard for the Kubernetes API Server metrics.

Note

The Kubernetes API Server exposes gauge, counter, histogram, and summary metrics. In this release of Prometheus metrics support, CloudWatch imports only the metrics with gauge and counter types.

To start collecting Prometheus Kubernetes API Server metrics in CloudWatch

1. Download the latest version of the `prometheus-eks.yaml` or `prometheus-k8s.yaml` file by entering one of the following commands. For an Amazon EKS cluster, enter the following command:

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

For a Kubernetes cluster running on an Amazon EC2 instance, enter the following command:

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
```

2. Open the file with a text editor, find the `prometheus-config` section, and add the following section inside of that section. Then save the changes:

```
# Scrape config for API servers
- job_name: 'kubernetes-apiservers'
  kubernetes_sd_configs:
    - role: endpoints
      namespaces:
        names:
          - default
  scheme: https
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    insecure_skip_verify: true
  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
  relabel_configs:
    - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
      action: keep
      regex: kubernetes;https
    - action: replace
      source_labels:
        - __meta_kubernetes_namespace
      target_label: Namespace
    - action: replace
      source_labels:
        - __meta_kubernetes_service_name
      target_label: Service
```

3. While you still have the YAML file open in the text editor, find the `cwagentconfig.json` section. Add the following subsection and save the changes. This section whitelists the KPI service metrics. Three types of KPI Server metrics are whitelisted:

- API Server request metrics
- API Server registration controller metrics
- ETCD helper cache metrics

```
{
  "source_labels": ["job"],
  "label_matcher": "^kubernetes-apiservers$",
  "dimensions": [["ClusterName", "Service"]],
  "metric_selectors": [
    "^etcd_helper_cache_entry_total$",
    "^etcd_helper_cache_hit_total$",
    "^etcd_helper_cache_miss_total$",
    "^APIServiceRegistrationController_depth$",
    "^APIServiceRegistrationController_adds$"
  ]
},
```

```
{
  "source_labels": ["job", "code"],
  "label_matcher": "^kubernetes-apiservers;2[0-9]{2}$",
  "dimensions": [{"ClusterName", "Service", "code"}],
  "metric_selectors": [
    "^apiserver_request_count$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^kubernetes-apiservers",
  "dimensions": [{"ClusterName", "Service"}],
  "metric_selectors": [
    "^apiserver_request_count$"
  ]
}
```

4. If you already have the CloudWatch agent with Prometheus support deployed in the cluster, you must delete it by entering the following command:

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

5. Deploy the CloudWatch agent with your updated configuration by entering one of the following commands. For an Amazon EKS cluster, enter:

```
kubectl apply -f prometheus-eks.yaml
```

For a Kubernetes cluster, enter this command:

```
kubectl apply -f prometheus-k8s.yaml
```

Once you have done this, you should see a new log stream named **kubernetes-apiservers** in the **/aws/containerinsights/*cluster_name*/prometheus** log group. This log stream should include log events with an embedded metric format definition like the following:

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Name": "apiserver_request_count"
        }
      ],
      "Dimensions": [
        [
          "ClusterName",
          "Service",
          "code"
        ]
      ],
      "Namespace": "ContainerInsights/Prometheus"
    },
    {
      "Metrics": [
        {
          "Name": "apiserver_request_count"
        }
      ],
      "Dimensions": [
        [
```

```
        "ClusterName",
        "Service"
    ],
    "Namespace": "ContainerInsights/Prometheus"
}
],
"ClusterName": "my-cluster-name",
"Namespace": "default",
"Service": "kubernetes",
"Timestamp": "1589182069416",
"Version": "0",
"apiserver_request_count": 0,
"apiserver_request_total": 0,
"code": "200",
"component": "apiserver",
"contentType": "application/json",
"group": "autoscaling",
"instance": "192.0.2.0:443",
"job": "kubernetes-apiservers",
"prom_metric_type": "counter",
"resource": "horizontalpodautoscalers",
"scope": "namespace",
"verb": "LIST",
"version": "v1"
}
```

You can view your metrics in the CloudWatch console in the **ContainerInsights/Prometheus** namespace. You can also optionally create a CloudWatch dashboard for your Prometheus Kubernetes KPI Server metrics.

(Optional) Creating a Dashboard for Kubernetes KPI Server metrics

To see Kubernetes API Server metrics in your dashboard, you must have first completed the steps in the previous sections to start collecting these metrics in CloudWatch.

To create a dashboard for Kubernetes KPI Server metrics

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Make sure you have the correct AWS Region selected.
3. In the navigation pane, choose **Dashboards**.
4. Choose **Create Dashboard**. Enter a name for the new dashboard, and choose **Create dashboard**.
5. In **Add to this dashboard**, choose **Cancel**.
6. Choose **Actions, View/edit source**.
7. Download the following JSON file: [Kubernetes API Dashboard source](#).
8. Open the JSON file that you downloaded with a text editor, and make the following changes:
 - Replace all the <YOUR_CLUSTER-NAME> strings with the exact name of your cluster. Make sure not to add whitespaces before or after the text.
 - Replace all the <YOUR_REGION> strings with the name of the Region where the metrics are collected. For example `us-west-2`. Be sure not to add whitespaces before or after the text.
9. Copy the entire JSON blob and paste it into the text box in the CloudWatch console, replacing what is already in the box.
10. Choose **Update, Save dashboard**.

(Optional) Set Up Sample Containerized Workloads for Prometheus Metric Testing

To test the Prometheus metric support in CloudWatch Container Insights, you can set up one or more of the following containerized workloads. The CloudWatch agent with Prometheus support automatically collects metrics from each of these workloads. To see the metrics that are collected by default, see [Prometheus Metrics Collected by the CloudWatch Agent \(p. 224\)](#).

Before you can install any of these workloads, you must install Helm 3.x by entering the following commands:

```
brew install helm
helm repo add stable https://kubernetes-charts.storage.googleapis.com
```

For more information, see [Helm](#).

Topics

- [\(Optional\) Set Up AWS App Mesh \(p. 215\)](#)
- [\(Optional\) Set Up NGINX with Sample Traffic \(p. 218\)](#)
- [\(Optional\) Set Up memcached with a Metric Exporter \(p. 219\)](#)
- [\(Optional\) Set Up Java/JMX \(p. 220\)](#)
- [\(Optional\) Set Up HAProxy with a Metric Exporter \(p. 223\)](#)

(Optional) Set Up AWS App Mesh

The beta of Prometheus support in CloudWatch Container Insights supports AWS App Mesh. This section explains how to set up App Mesh.

CloudWatch Container Insights can also collect App Mesh Envoy Access Logs. For more information, see [\(Optional\) Enable App Mesh Envoy Access Logs \(p. 174\)](#).

Configure IAM Permissions

You must add the **AWSAppMeshFullAccess** policy to the IAM role for your Amazon EKS or Kubernetes node group. On Amazon EKS, this node group name looks similar to `eksctl-integ-test-eks-prometheus-NodeInstanceRole-ABCDEFGHIJKL`. On Kubernetes, it might look similar to `nodes.integ-test-kops-prometheus.k8s.local`.

Install App Mesh

Follow these steps to install App Mesh.

To install App Mesh for testing with CloudWatch Container Insights Prometheus support

1. Enter the following command to add the Helm repo. This repo works on both Amazon EKS and Kubernetes clusters.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Enter the following command to apply the App Mesh custom resource definition:

```
kubectl apply -f https://raw.githubusercontent.com/aws/eks-charts/master/stable/appmesh-controller/crds/crds.yaml
```

3. Set the environment variables for the App Mesh workload. The following lines are an example, you can choose your own environment variables:

```
APPMESH_NAME=appmesh-sample
APPMESH_SYSTEM_NAMESPACE=appmesh-system-sample
APPMESH_SAMPLE_TRAFFIC_NAMESPACE=appmesh-sample-traffic-test
```

4. Install the App Mesh custom resource definition controller by entering the following commands:

```
kubectl create namespace $APPMESH_SYSTEM_NAMESPACE
helm upgrade -i appmesh-controller eks/appmesh-controller \
--namespace $APPMESH_SYSTEM_NAMESPACE
```

5. Install the App Mesh injector by entering the following command:

```
helm upgrade -i appmesh-inject eks/appmesh-inject \
--namespace $APPMESH_SYSTEM_NAMESPACE \
--set mesh.create=true \
--set mesh.name=$APPMESH_NAME \
--set stats.tagsEnabled=true
```

6. (Optional) To confirm that the sample service mesh has been created, Open the App Mesh console at <https://console.aws.amazon.com/app-mesh/>. Set the console to the AWS Region where your cluster is running, and choose **Meshes**. You should see a new mesh named `appmesh-sample`.
7. Create App Mesh testing traffic by entering the following command:

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/
cwagent-prometheus/sample_traffic/appmesh/appmesh_traffic_sample.yaml |
sed "s/{{appmeshname}}/$APPMESH_NAME/g" |
sed "s/{{namespace}}/$APPMESH_SAMPLE_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

8. (Optional) Install a second App Mesh workload service to generate 404 and 503 response codes by entering the following command:

```
APPMESH_SAMPLE_ERROR_TRAFFIC_NAMESPACE=appmesh-sample-errortraffic-test
```

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/
cwagent-prometheus/sample_traffic/appmesh/appmesh_traffic_sample_4xx5xx.yaml |
sed "s/{{appmeshname}}/$APPMESH_NAME/g" |
sed "s/{{namespace}}/$APPMESH_SAMPLE_ERROR_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

9. Find the name of the traffic generator by entering the following command:

```
kubectl get pod -n $APPMESH_SAMPLE_TRAFFIC_NAMESPACE
```

The traffic generator is in the `NAME` column and starts with `traffic-generator`

10. Find the name of the virtual service by entering the following command:

```
kubectl get virtualservice.appmesh.k8s.aws -n $APPMESH_SAMPLE_TRAFFIC_NAMESPACE
```

11. Remote exec into the traffic generator pod by entering the following command. Substitute *traffic-generator-name* with the name of the traffic generator that you found in step 9.


```
kubectl exec -ti traffic-generator-name bash -n $APPMESH_SAMPLE_TRAFFIC_NAMESPACE
```

12. Run the curl command multiple times, using the name of the virtual service that you found in step 10.

```
curl virtual_service_name:9080;echo
```

13. Exit the pod by entering the following command:

```
Exit
```

14. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
15. In the AWS Region where your cluster is running, choose **Metrics**. The App Mesh meetrics are in the **ContainerInsights/Prometheus** namespace.
16. To see the CloudWatch Logs events, choose **Log Groups** in the navigation pane. The events are in the log group **/aws/containerinsights/*your_cluster_name*/prometheus**, in the log stream **kubernetes-pod-appmesh-envoyappmesh**.

Troubleshooting App Mesh

The following can help you troubleshoot your App Mesh testing setup.

Pod Initialized in Pending Status

This example setup creates 20 pods. You can check whether any pods are still pending by entering the following commands:

```
kubectl get pod -n $APPMESH_SYSTEM_NAMESPACE  
kubectl get pod -n $APPMESH_SAMPLE_TRAFFIC_NAMESPACE  
kubectl get pod -n $APPMESH_SAMPLE_ERROR_TRAFFIC_NAMESPACE
```

If any pods are pending, your cluster might not have enough CPU or memory capacity to provision the required pods. To solve this, provision a new EC2 instance to your cluster.

Error Running Load Generator

You might see the following message when you run the load generator:

```
curl: (7) Failed to connect to jazz.appmesh-sample-traffic-test.svc.cluster.local port  
9080: Connection refused
```

This could be that the App Mesh IAM policy is not set up on the node group. To solve this, add the **AWSAppMeshFullAccess** policy to the IAM role for your Amazon EKS or Kubernetes node group. On Amazon EKS, this node group name looks similar to **eksctl-integ-test-eks-prometheus-NodeInstanceRole-ABCDEFHIJKL**. On Kubernetes, it might look similar to **nodes.integ-test-kops-prometheus.k8s.local**.

Once you have done this, confirm that the sample service mesh has been created. Open the App Mesh console at <https://console.aws.amazon.com/app-mesh/>. Set the console to the AWS Region where your cluster is running, and choose **Meshes**. You should see a new mesh named **appmesh-sample**.

Deleting The App Mesh Test Environment

When you have finished using App Mesh and the traffic generator for testing, use the following commands to delete the unnecessary resources.

Delete the sample traffic by entering the following command:

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/  
prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-  
prometheus/sample_traffic/appmesh/appmesh_traffic_sample_4xx5xx.yaml |  
sed "s/{{appmeshname}}/$APPMESH_NAME/g" |  
sed "s/{{namespace}}/$APPMESH_SAMPLE_ERROR_TRAFFIC_NAMESPACE/g" |  
kubectl delete -f -curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-  
container-insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/  
service/cwagent-prometheus/sample_traffic/appmesh/appmesh_traffic_sample.yaml |  
sed "s/{{appmeshname}}/$APPMESH_NAME/g" |  
sed "s/{{namespace}}/$APPMESH_SAMPLE_TRAFFIC_NAMESPACE/g" |  
kubectl delete -f -
```

Delete the App Mesh controller and injector by entering the following command:

```
kubectl delete namespace appmesh-system-sample
```

Delete the App Mesh controller cluster-level objects:

```
kubectl delete clusterrole appmesh-controller  
kubectl delete clusterrolebinding appmesh-controller
```

Delete the App Mesh injector cluster level objects:

```
kubectl delete clusterrole appmesh-inject  
kubectl delete clusterrolebinding appmesh-inject  
kubectl delete MutatingWebhookConfiguration appmesh-inject
```

(Optional) Set Up NGINX with Sample Traffic

NGINX is a web server that can also be used as a load balancer and reverse proxy. For more information, see [NGINX](#).

To install NGINX with a sample traffic service to test Container Insights Prometheus support

1. Enter the following commands:

```
kubectl create namespace nginx-ingress-sample  
  
helm install stable/nginx-ingress --generate-name --version 1.33.5 \  
--namespace nginx-ingress-sample \  
--set controller.metrics.enabled=true \  
--set controller.metrics.service.annotations."prometheus\.io/port"="10254" \  
--set controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

2. Check whether the services started correctly by entering the following command:

```
kubectl get service -n nginx-ingress-sample
```

The output of this command should display several columns, including an `EXTERNAL-IP` column.

3. Set an `EXTERNAL-IP` variable to the value of the `EXTERNAL-IP` column in the row of the NGINX ingress controller.

```
EXTERNAL_IP=your-nginx-controller-external-ip
```

4. Start some sample NGINX traffic by entering the following command.

```
SAMPLE_TRAFFIC_NAMESPACE=nginx-sample-traffic
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/prometheus-beta/k8s-deployment-manifest-templates/deployment-mode/service/
cwagent-prometheus/sample_traffic/nginx-traffic/nginx-traffic-sample.yaml |
sed "s/{external_ip}}/$EXTERNAL_IP/g" |
sed "s/{namespace}}/$SAMPLE_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

5. Enter the following command to confirm that all three pods are in the Running status.

```
kubectl get pod -n $SAMPLE_TRAFFIC_NAMESPACE
```

If they are running, you should soon see metrics in the **ContainerInsights/Prometheus** namespace.

To uninstall NGINX and the sample traffic application

1. Delete the sample traffic service by entering the following command:

```
kubectl delete namespace $SAMPLE_TRAFFIC_NAMESPACE
```

2. Delete the NGINX egress. First, find the Helm release name. For example, if the service name is `nginx-ingress-1583277500-controller`, the Helm release name is `nginx-ingress-1583277500`

Then enter the following commands:

```
helm uninstall release-name --namespace nginx-ingress-sample
kubectl delete namespace nginx-ingress-sample $SAMPLE_TRAFFIC_NAMESPACE
```

(Optional) Set Up memcached with a Metric Exporter

memcached is an open-source memory object caching system. For more information, see [What is Memcached?](#).

To install memcached with a metric exporter to test Container Insights Prometheus support

1. Enter the following command to create a new namespace:

```
kubectl create namespace memcached-sample
```

2. Enter the following command to install Memcached

```
helm install my-memcached stable/memcached --namespace memcached-sample \
--set metrics.enabled=true \
--set-string serviceAnnotations.prometheus\\.io/port="9150" \
--set-string serviceAnnotations.prometheus\\.io/scrape="true"
```

3. Enter the following command to confirm the annotation of the running service:

```
kubectl describe service my-memcached -n memcached-sample
```

You should see the following two annotations:

```
Annotations:  prometheus.io/port: 9150
```

```
prometheus.io/scrape: true
```

To uninstall memcached

- Enter the following commands:

```
helm uninstall my-memcached --namespace memcached-sample  
kubectl delete namespace memcached-sample
```

(Optional) Set Up Java/JMX

JMX Exporter is an official Prometheus exporter that can scrape and expose JMX mBeans as Prometheus metrics. For more information, see [prometheus/jmx_exporter](#).

Container Insights can collect predefined Prometheus metrics from Java Virtual Machine (JVM), Java, and Tomcat (Catalina) using the JMX Exporter.

The CloudWatch agent with Prometheus support scrapes the Java/JMX Prometheus metrics from `http://CLUSTER_IP:9404/metrics` on each pod in an Amazon EKS or Kubernetes cluster. You can configure the JMX Exporter to expose the metrics on a different port or `metrics_path`. If you do change the port or path, update the default `jmx_scrape_config` in the CloudWatch agent config map. Run the following command to get the current CloudWatch agent Prometheus configuration:

```
kubectl describe cm prometheus-config -n amazon-cloudwatch
```

The fields to change are the `/metrics` and `regex: '.*:9404$'` fields, as highlighted in the following example.

```
job_name: 'kubernetes-jmx-pod'  
sample_limit: 10000  
metrics_path: /metrics  
kubernetes_sd_configs:  
- role: pod  
relabel_configs:  
- source_labels: [__address__]  
  action: keep  
  regex: '.*:9404$'  
- action: replace  
  regex: (.+)  
  source_labels:
```

Next, build a docker image. The following sections provide two example dockerfiles.

When you have built the image, load it into Amazon EKS or Kubernetes, and then run the following command to verify that Prometheus metrics are exposed by `JMX_EXPORTER` on port 9404. Replace `$JAR_SAMPLE_TRAFFIC_POD` with the running pod name and replace `$JAR_SAMPLE_TRAFFIC_NAMESPACE` with your application namespace.

```
kubectl exec $JAR_SAMPLE_TRAFFIC_POD -n $JARCAT_SAMPLE_TRAFFIC_NAMESPACE -- curl http://localhost:9404
```

Example: Apache Tomcat Docker Image with Prometheus Metrics

Apache Tomcat server exposes JMX mBeans by default. You can integrate JMX Exporter with Tomcat to expose JMX mBeans as Prometheus metrics. The following example dockerfile shows the steps to build a testing image:

```
# From Tomcat 9.0 JDK8 OpenJDK
FROM tomcat:9.0-jdk8-openjdk

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter
COPY ./setenv.sh /usr/local/tomcat/bin
COPY your web application.war /usr/local/tomcat/webapps/

RUN chmod o+x /usr/local/tomcat/bin/setenv.sh

ENTRYPOINT ["catalina.sh", "run"]
```

The following list explains the four COPY lines in this dockerfile.

- Download the latest JMX Exporter jar file from https://github.com/prometheus/jmx_exporter.
- config.yaml is the JMX Exporter configuration file. For more information, see https://github.com/prometheus/jmx_exporter#Configuration.

Here is a sample configuration file for Java and Tomcat:

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([a-zA-Z0-9+&@#/%?=-_!|.:,;]*[-
a-zA-Z0-9+&@#/%?=-_!|.:,;]*), name=([-a-zA-Z0-9+/%?=-_!|.:,;]*), J2EEApplication=none,
J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name=\"(\w+-\w+)-(\d+)\"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE
```

```
- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=-_!:.;,]*[-a-zA-Z0-9+&@#/%?=-_!:.], context=([-a-zA-Z0-9+/%?=-_!:.]*><(processingTime|sessionCounter|rejectedSessions|expiredSessions))'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `setenv.sh` is a Tomcat startup script to start the JMX exporter along with Tomcat and expose Prometheus metrics on port 9404 of the localhost. It also provides the JMX Exporter with the `config.yaml` file path.

```
$ cat setenv.sh
export JAVA_OPTS="-javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml $JAVA_OPTS"
```

- your `web application.war` is your web application war file to be loaded by Tomcat.

Build a docker image with this configuration and upload it to an image repository.

Example: Java Jar Application Docker Image with Prometheus Metrics

The following example dockerfile shows the steps to build a testing image:

```
# Alpine Linux with OpenJDK JRE
FROM openjdk:8-jre-alpine

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./SampleJavaApplication-1.0-SNAPSHOT.jar /opt/jmx_exporter
COPY ./start_exporter_example.sh /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter

RUN chmod -R o+x /opt/jmx_exporter
RUN apk add curl

ENTRYPOINT exec /opt/jmx_exporter/start_exporter_example.sh
```

The following list explains the four `COPY` lines in this dockerfile.

- Download the latest JMX Exporter jar file from https://github.com/prometheus/jmx_exporter.
- `config.yaml` is the JMX Exporter configuration file. For more information, see https://github.com/prometheus/jmx_exporter#Configuration.

Here is a sample configuration file for Java and Tomcat:

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE
```

```
- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\\w+-\\w+)-(\\d+)\"><>(\\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*), name=([-a-zA-Z0-9+&@#/%?=-_!|.:,;]*), J2EEApplication=none, J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name=\"(\\w+-\\w+)-(\\d+)\"><>(currentThreadCount|currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*), context=([-a-zA-Z0-9+&@#/%?=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*[-a-zA-Z0-9+&@#/%=-_!|.:,;]*)><>(processingTime|sessionCounter|rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `start_exporter_example.sh` is the script to start the JAR application with the Prometheus metrics exported. It also provides the JMX Exporter with the `config.yaml` file path.

```
$ cat start_exporter_example.sh
java -javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml -cp /opt/jmx_exporter/SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App
```

- `SampleJavaApplication-1.0-SNAPSHOT.jar` is the sample Java application jar file. Replace it with the Java application that you want to monitor.

Build a docker image with this configuration and upload it to an image repository.

(Optional) Set Up HAProxy with a Metric Exporter

HAProxy is an open-source proxy application. For more information, see [HAProxy](#).

To install HAProxy with a metric exporter to test Container Insights Prometheus support

1. Enter the following command to add the Helm incubator repo:

```
helm repo add incubator https://kubernetes-charts-incubator.storage.googleapis.com
```

2. Enter the following command to create a new namespace:

```
kubectl create namespace haproxy-ingress-sample
```

3. Enter the following commands to install HAProxy:

```
helm install haproxy incubator/haproxy-ingress \
--namespace haproxy-ingress-sample \
--set controller.stats.enabled=true \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="9101" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

4. Enter the following command to confirm the annotation of the service:

```
kubectl describe service haproxy-haproxy-ingress-controller-metrics -n haproxy-ingress-sample
```

You should see the following annotations.

```
Annotations:  prometheus.io/port: 9101
              prometheus.io/scrape: true
```

To uninstall HAProxy

- Enter the following commands:

```
helm uninstall haproxy --namespace haproxy-ingress-sample
kubectl delete namespace haproxy-ingress-sample
```

Prometheus Metrics Collected by the CloudWatch Agent

Support for Prometheus metrics is in beta. The beta is open to all AWS accounts and you do not need to request access. Features may be added or changed before announcing General Availability. Don't hesitate to contact us with any feedback or let us know if you would like to be informed when updates are made by emailing us at containerinsightsfeedback@amazon.com

The CloudWatch agent with Prometheus support automatically collects metrics from several services and workloads. These metrics are listed in the following sections. You can also configure the agent to collect other Prometheus metrics from other applications and services.

All Prometheus metrics are collected in the **ContainerInsights/Prometheus** namespace.

Topics

- [Prometheus Metrics for App Mesh \(p. 225\)](#)
- [Prometheus Metrics for NGINX \(p. 226\)](#)

- [Prometheus Metrics for memcached \(p. 226\)](#)
- [Prometheus Metrics for Java/JMX \(p. 227\)](#)
- [Prometheus Metrics for HAProxy \(p. 228\)](#)

Prometheus Metrics for App Mesh

The following metrics are automatically collected from App Mesh.

CloudWatch Container Insights can also collect App Mesh Envoy Access Logs. For more information, see [\(Optional\) Enable App Mesh Envoy Access Logs \(p. 174\)](#).

Metric Name	Dimensions	
envoy_http_downstream_requests_total	ClusterName, Namespace	
envoy_http_downstream_bytes_total	ClusterName, Namespace	
envoy_http_downstream_bytes_received	ClusterName, Namespace, envoy_http_conn_manager_prefix, envoy_response_code_class	
envoy_cluster_upstream_requests_total	ClusterName, Namespace	
envoy_cluster_upstream_bytes_total	ClusterName, Namespace	
envoy_cluster_member_count	ClusterName, Namespace	
envoy_cluster_member_healthy	ClusterName, Namespace	
envoy_server_memory_allocated	ClusterName, Namespace	
envoy_server_memory_free	ClusterName, Namespace	
envoy_cluster_upstream_requests_out	ClusterName, Namespace	
envoy_cluster_upstream_requests_reject	ClusterName, Namespace	
envoy_cluster_upstream_requests_slow	ClusterName, Namespace	
envoy_cluster_upstream_requests_timeout	ClusterName, Namespace	
envoy_cluster_upstream_requests_out	ClusterName, Namespace	
envoy_cluster_upstream_requests_timeout	ClusterName, Namespace	
envoy_cluster_upstream_requests_with_active_rq	ClusterName, Namespace	
envoy_cluster_upstream_requests_active_rq	ClusterName, Namespace	
envoy_cluster_upstream_requests_code	ClusterName, Namespace	
envoy_cluster_upstream_requests_pauses_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_requests_resumed_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_requests_pauses_wrote_total	ClusterName, Namespace	
envoy_cluster_upstream_requests_resumed_wrote_total	ClusterName, Namespace	
envoy_cluster_upstream_requests_timeout	ClusterName, Namespace	

Metric Name	Dimensions	
envoy_cluster_upstream_requests	ClusterName, Namespace	
envoy_cluster_upstream_requests_slow	ClusterName, Namespace	
envoy_server_live	ClusterName, Namespace	
envoy_server_uptime	ClusterName, Namespace	

Prometheus Metrics for NGINX

The following metrics are automatically collected from NGINX.

Metric Name	Dimensions	
nginx_ingress_controller_seconds_total	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	
nginx_ingress_controller_status	ClusterName, Namespace, Service	

Prometheus Metrics for memcached

The following metrics are automatically collected from memcached.

Metric Name	Dimensions	
memcached_current_items	ClusterName, Namespace, Service	
memcached_current_connections	ClusterName, Namespace, Service	
memcached_limit_bytes	ClusterName, Namespace, Service	
memcached_current_bytes	ClusterName, Namespace, Service	
memcached_written_bytes	ClusterName, Namespace, Service	
memcached_read_bytes	ClusterName, Namespace, Service	
memcached_items_evicted	ClusterName, Namespace, Service	
memcached_items_rejected	ClusterName, Namespace, Service	
memcached_commands_total	ClusterName, Namespace, Service	
	ClusterName, Namespace, Service, command	

Metric Name	Dimensions	
	ClusterName, Namespace, Service, status, command	

Prometheus Metrics for Java/JMX

Container Insights can collect the following predefined Prometheus metrics from the Java Virtual Machine (JVM), Java, and Tomcat (Catalina) using the JMX Exporter. For more information, see [prometheus/jmx_exporter](#) on Github.

Java/JMX

Metric Name	Dimensions	
jvm_classes_loaded	ClusterName, Namespace	
jvm_threads_current	ClusterName, Namespace	
jvm_threads_daemon	ClusterName, Namespace	
java_lang_operatingsystem_memorysize	ClusterName, Namespace	
java_lang_operatingsystem_heapsize	ClusterName, Namespace	
java_lang_operatingsystem_nonheapsize	ClusterName, Namespace	
java_lang_operatingsystem_memorysize	ClusterName, Namespace	
java_lang_operatingsystem_heapsize	ClusterName, Namespace	
java_lang_operatingsystem_nonheapsize	ClusterName, Namespace	
java_lang_operatingsystem_memorysize	ClusterName, Namespace	
java_lang_operatingsystem_heapsize	ClusterName, Namespace	
java_lang_operatingsystem_nonheapsize	ClusterName, Namespace	
java_lang_operatingsystem_memorysize	ClusterName, Namespace	
java_lang_operatingsystem_heapsize	ClusterName, Namespace	
java_lang_operatingsystem_nonheapsize	ClusterName, Namespace	
jvm_memory_bytes_used	ClusterName, Namespace, area	
jvm_memory_pool_bytes_used	ClusterName, Namespace, pool	

Note

The values of the area dimension can be heap or nonheap.

The values of the pool dimension can be Tenured Gen, Compress Class Space, Survivor Space, Eden Space, Code Cache, or Metaspace.

Tomcat/JMX

In addition to the Java/JMX metrics in the previous table, the following metrics are also collected for the Tomcat workload.

Metric Name	Dimensions	
catalina_manager_active_sessions	ClusterName, Namespace	
catalina_manager_rejected_sessions	ClusterName, Namespace	

Metric Name	Dimensions	
catalina_globalrequest_received	ClusterName, Namespace	
catalina_globalrequest_processed	ClusterName, Namespace	
catalina_globalrequest_count	ClusterName, Namespace	
catalina_globalrequest_count	ClusterName, Namespace	
catalina_globalrequest_time	ClusterName, Namespace	

Prometheus Metrics for HAProxy

The following metrics are automatically collected from HAProxy.

Metric Name	Dimensions	
haproxy_backend_bytes	ClusterName, Namespace, Service	
haproxy_backend_bytes	ClusterName, Namespace, Service	
haproxy_backend_connections	ClusterName, Namespace, Service	
haproxy_backend_current	ClusterName, Namespace, Service	
haproxy_backend_current	ClusterName, Namespace, Service	
haproxy_backend_current	ClusterName, Namespace, Service	
haproxy_backend_http_requests	ClusterName, Namespace, Service, code, backend	
haproxy_backend_limit	ClusterName, Namespace, Service	
haproxy_backend_max	ClusterName, Namespace, Service	
haproxy_backend_mex	ClusterName, Namespace, Service	
haproxy_backend_mex	ClusterName, Namespace, Service	
haproxy_backend_redis	ClusterName, Namespace, Service	
haproxy_backend_response	ClusterName, Namespace, Service	
haproxy_backend_retry_warnings_total	ClusterName, Namespace, Service	
haproxy_backend_up	ClusterName, Namespace, Service	
haproxy_backend_weight	ClusterName, Namespace, Service	
haproxy_backend_bytes	ClusterName, Namespace, Service	
haproxy_frontend_bytes	ClusterName, Namespace, Service	
haproxy_frontend_bytes	ClusterName, Namespace, Service	
haproxy_frontend_connections	ClusterName, Namespace, Service	
haproxy_frontend_current	ClusterName, Namespace, Service	

Metric Name	Dimensions	
haproxy_frontend_curr	ClusterName, Namespace, Service	
haproxy_frontend_http	ClusterName, Namespace, Service	
haproxy_frontend_http	ClusterName, Namespace, Service, code, frontend	
haproxy_frontend_lim	ClusterName, Namespace, Service	
haproxy_frontend_lim	ClusterName, Namespace, Service	
haproxy_frontend_max	ClusterName, Namespace, Service	
haproxy_frontend_max	ClusterName, Namespace, Service	
haproxy_frontend_req	ClusterName, Namespace, Service	
haproxy_frontend_req	ClusterName, Namespace, Service	

Note

The values of the code dimension can be 1xx, 2xx, 3xx, 4xx, 5xx, or other.

The values of the backend dimension can be error413, error495, error496, error503noendpoints, http-default-backend, upstream-default-backend, or other values.

The values of the frontend dimension can be error503noendpoints, httpfront-default-backend, state/code>, healthzor other values.

Viewing Your Prometheus Metrics

You can monitor and alarm on all your Prometheus metrics including the curated pre-aggregated metrics from App Mesh, NGINX, Java/JMX, Memcached, and HAProxy, and any other manually configured Prometheus exporter you may have added. For more information about collecting metrics from other Prometheus exporters, see [Tutorial for Adding a New Prometheus Scrape Target: Prometheus KPI Server Metrics \(p. 211\)](#).

Container Insights also provides automatic dashboards for App Mesh and NGINX. You can also access these dashboards as custom dashboards that you can configure in your environment for App Mesh and NGINX.

To see all your Prometheus metrics

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. In the list of namespaces, choose **ContainerInsights/Prometheus**.
4. Choose one of the sets of dimensions in the following list. Then select the checkbox next to the metrics that you want to see.

To see pre-built reports on your Prometheus metrics

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Performance Monitoring**.
3. In the drop-down box near the top of the page, choose **Prometheus AppMesh** or **Prometheus NGINX**.

In the other drop-down box, choose a cluster to view

We have also provided custom dashboards for NGINX, App Mesh, and Java/JMX.

To use a custom dashboard that Amazon has provided

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Dashboards**.
3. Choose **Create Dashboard**. Enter a name for the new dashboard, and choose **Create dashboard**.
4. In **Add to this dashboard**, choose **Cancel**.
5. Choose **Actions, View/edit source**.
6. Download one of the following JSON files:
 - [NGINX custom dashboard source on Github](#).
 - [App Mesh custom dashboard source on Github](#).
 - [Java/JMX custom dashboard source on Github](#).
7. Open the JSON file that you downloaded with a text editor, and make the following changes:
 - Replace all the <CLUSTER-NAME> strings with the exact name of your cluster. Make sure not to add whitespaces before or after the text.
 - (Optional) If your cluster is in a Region other than US East (N. Virginia), replace all the `us-east-1` strings with the AWS Region where your cluster is running. Make sure not to add whitespaces before or after the text.
8. Copy the entire JSON blob and paste it into the text box in the CloudWatch console, replacing what is already in the box.
9. Choose **Update, Save dashboard**.

Prometheus Metrics Troubleshooting

This section provides help for troubleshooting your Prometheus metrics setup.

General Troubleshooting Steps

To confirm that the CloudWatch agent is running, enter the following command.

```
kubectl get pod -n amazon-cloudwatch
```

The output should include a row with `cwagent-prometheus-id` in the `NAME` column and `Running` in the `STATUS` column.

To display details about the running pod, enter the following command. Replace `pod-name` with the complete name of your pod that has a name that starts with `cw-agent-prometheus`.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

If you have CloudWatch Container Insights installed, you can use CloudWatch Logs Insights to query the logs from the CloudWatch agent collecting the Prometheus metrics.

To query the application logs

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **CloudWatch Logs Insights**.
3. Select the log group for the application logs, `/aws/containerinsights/cluster-name/application`.
4. Replace the search query expression with the following query, and choose **Run query**

```
fields ispresent(kubernetes.pod_name) as haskubernetes_pod_name, stream,  
kubernetes.pod_name, log |  
filter haskubernetes_pod_name and kubernetes.pod_name like /cwagent-prometheus
```

You can also confirm that Prometheus metrics and metadata are being ingested as CloudWatch Logs events.

To confirm that Prometheus data is being ingested

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **CloudWatch Logs Insights**.
3. Select the **/aws/containerinsights/*cluster-name*/prometheus**
4. Replace the search query expression with the following query, and choose **Run query**

```
fields @timestamp, @message | sort @timestamp desc | limit 20
```

Logging Dropped Prometheus Metrics

This beta release does not collect Prometheus metrics of the histogram or summary types. You can use the CloudWatch agent to check whether any Prometheus metrics are being dropped because they are one of these types. You can also log a list of the first 500 Prometheus metrics that are dropped and not sent to CloudWatch because they are histogram or summary metrics.

To see whether any metrics are being dropped, enter the following command:

```
kubectll logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

If any metrics are being dropped, you will see the following lines in the `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` file.

```
I! Drop Prometheus metrics with unsupported types. Only Gauge and Counter are supported.  
I! Please enable CWAgent debug mode to view the first 500 dropped metrics
```

If you see those lines and want to know what metrics are being dropped, use the following steps.

To log a list of dropped Prometheus metrics

1. Change the CloudWatch agent to debug mode by adding the following bold lines to your `prometheus-eks.yaml` or `prometheus-k8s.yaml` file, and save the file.

```
{  
  "agent": {  
    "debug": true  
  },  
}
```

This section of the file should then look like this:

```
cwagentconfig.json: |  
{  
  "agent": {  
    "debug": true  
  },  
}
```

```
"logs": {  
  "metrics_collected": {
```

2. Reinstall the CloudWatch agent to enable debug mode by entering the following commands:

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch  
kubectl apply -f prometheus.yaml
```

The dropped metrics are logged in the CloudWatch agent pod.

3. To retrieve the logs from the CloudWatch agent pod, enter the following command:

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

Or, if you have Container Insights FluentD logging installed, the logs are also saved in the CloudWatch Logs log group **/aws/containerinsights/*cluster_name*/application**.

To query these logs, you can follow the steps for querying the application logs in [General Troubleshooting Steps](#) (p. 230).

Where are the Prometheus Metrics Ingested as CloudWatch Logs Log Events?

The CloudWatch agent creates a log stream for each Prometheus scrape job configuration. For example, in the `prometheus-eks.yaml` and `prometheus-k8s.yaml` files, the line `job_name: 'kubernetes-pod-appmesh-envoy'` scrapes App Mesh metrics. The Prometheus target is defined as `kubernetes-pod-appmesh-envoy`. So all App Mesh Prometheus metrics are ingested as CloudWatch Logs events in the log stream **kubernetes-pod-appmesh-envoy** under the log group named **/aws/containerinsights/*cluster-name*/Prometheus**.

I Don't See Prometheus Metrics in CloudWatch Metrics

First, make sure that the Prometheus metrics are ingested as log events in the log group **/aws/containerinsights/*cluster-name*/Prometheus**. Use the information in [Where are the Prometheus Metrics Ingested as CloudWatch Logs Log Events?](#) (p. 232) to help you check the target log stream. If the log stream is not created or there are no new log events in the log stream, check the following:

- Check that the Prometheus metrics exporter endpoints are set up correctly
- Check that the Prometheus scraping configurations in the `config map: cwagent-prometheus` section of the CloudWatch agent YAML file is correct. The configuration should be the same as it would be in a Prometheus configuration file. For more information, see [<scrape_config>](#) in the Prometheus documentation.

If the Prometheus metrics are ingested as log events correctly, check that the embedded metric format settings are added into the log events to generate the CloudWatch metrics.

```
"CloudWatchMetrics": [  
  {  
    "Metrics": [  
      {  
        "Name": "envoy_http_downstream_cx_destroy_remote_active_rq"  
      }  
    ],  
    "Dimensions": [  
      [  
        "ClusterName",
```



```
        "Namespace"
      ],
      "Namespace": "ContainerInsights/Prometheus"
    }
  ],
```

For more information about embedded metric format, see [Specification: Embedded Metric Format \(p. 388\)](#).

If there is no embedded metric format in the log events, check that the `metric_definitions` are configured correctly in the `config map: prometheus-cwagentconfig` section of the CloudWatch agent installation YAML file. For more information, see [Tutorial for Adding a New Prometheus Scrape Target: Prometheus KPI Server Metrics \(p. 211\)](#).

Troubleshooting Container Insights

The following sections can help if you're having trouble issues with Container Insights.

Failed Deployment on Amazon EKS or Kubernetes

If the agent doesn't deploy correctly on a Kubernetes cluster, try the following:

- Run the following command to get the list of pods.

```
kubectl get pods -n amazon-cloudwatch
```

- Run the following command and check the events at the bottom of the output.

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- Run the following command to check the logs.

```
kubectl logs pod-name -n amazon-cloudwatch
```

Unauthorized panic: Cannot retrieve cadvisor data from kubelet

If your deployment fails with the error `Unauthorized panic: Cannot retrieve cadvisor data from kubelet`, your kubelet might not have Webhook authorization mode enabled. This mode is required for Container Insights. For more information, see [Verify Prerequisites \(p. 165\)](#).

Deploying Container Insights on a Deleted and Re-Created Cluster

If you delete an existing cluster that does not have Container Insights enabled, and you re-create it with the same name, you can't enable Container Insights on this new cluster at the time you re-create it. You can enable it by re-creating it, and then entering the following command:

```
aws ecs update-cluster-settings --cluster myCICluster --settings  
name=containerInsights,value=enabled
```

Invalid Endpoint Error

If you see an error message similar to the following, check to make sure that you replaced all the placeholders such as *cluster-name* and *region-name* in the commands that you are using with the correct information for your deployment.

```
"log": "2020-04-02T08:36:16Z E! cloudwatchlogs: code: InvalidEndpointURL, message:
invalid endpoint uri, original error: &url.Error{Op:\"parse\", URL:\"https://logs.
{{region_name}}.amazonaws.com/\", Err:\"\"}, &awserr.baseError{code:\"InvalidEndpointURL
\", message:\"invalid endpoint uri\", errs:[]error{(*url.Error)(0xc0008723c0)}}\n",
```

Metrics Don't Appear in the Console

If you don't see any Container Insights metrics in the AWS Management Console, be sure that you have completed the setup of Container Insights. Metrics don't appear before Container Insights has been set up completely. For more information, see [Setting Up Container Insights \(p. 157\)](#).

CrashLoopBackoff Error on the CloudWatch Agent

If you see a `CrashLoopBackOff` error for the CloudWatch agent, make sure that your IAM permissions are set correctly. For more information, see [Verify Prerequisites \(p. 165\)](#).

CloudWatch Agent or FluentD Pod Stuck in Pending

If you have a CloudWatch agent or FluentD pod stuck in `Pending` or with a `FailedScheduling` error, determine if your nodes have enough compute resources based on the number of cores and amount of RAM required by the agents. Enter the following command to describe the pod:

```
kubectl describe pod cloudwatch-agent-85ppg -n amazon-cloudwatch
```

Building Your Own CloudWatch Agent Docker Image

You can build your own CloudWatch agent Docker image by referring to the Dockerfile located at <https://github.com/aws-samples/amazon-cloudwatch-container-insights/blob/master/cloudwatch-agent-dockerfile/Dockerfile>.

Deploying Other CloudWatch Agent Features in Your Containers

You can deploy additional monitoring features in your containers using the CloudWatch agent. These features include the following:

- **Embedded Metric Format**— For more information, see [Ingesting High-Cardinality Logs and Generating Metrics with CloudWatch Embedded Metric Format \(p. 387\)](#).
- **StatsD**— For more information, see [Retrieve Custom Metrics with StatsD \(p. 302\)](#).

Instructions and necessary files are located on GitHub at the following locations:

- For Amazon ECS containers, see [Example Amazon ECS task definitions based on deployment modes](#).
- For Amazon EKS and Kubernetes containers, see [Example Kubernetes YAML files based on deployment modes](#).

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

The unified CloudWatch agent enables you to do the following:

- Collect more system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances. The additional metrics that can be collected are listed in [Metrics Collected by the CloudWatch Agent \(p. 305\)](#).
- Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
- Retrieve custom metrics from your applications or services using the `StatsD` and `collectd` protocols. `StatsD` is supported on both Linux servers and servers running Windows Server. `collectd` is supported only on Linux servers.
- Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.

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

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

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

The steps in this section explain how to install the unified CloudWatch agent on Amazon EC2 instances and on-premises servers. For more information about the metrics that the CloudWatch agent can collect, see [Metrics Collected by the CloudWatch Agent \(p. 305\)](#).

Supported Operating Systems

The CloudWatch agent is supported on AMD64 architecture on the following operating systems:

- Amazon Linux version 2014.03.02 or later
- Amazon Linux 2
- Ubuntu Server versions 18.04, 16.04, and 14.04
- CentOS versions 7.6, 7.2, 7.0, 6.8, and 6.5
- Red Hat Enterprise Linux (RHEL) versions 8, 7.6, 7.5, 7.4, 7.2, 7.0, and 6.5
- Debian 8.0
- SUSE Linux Enterprise Server (SLES) version 12 and version 15
- 64-bit versions of Windows Server 2019, Windows Server 2016, Windows Server 2012, and Windows Server 2008 R2

The agent is supported on ARM64 architecture on the following operating systems:

- Amazon Linux 2
- Ubuntu Server version 18.04
- Red Hat Enterprise Linux (RHEL) version 7.6
- SUSE Linux Enterprise Server 15

Installation Process Overview

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

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

Contents

- [Installing the CloudWatch Agent \(p. 237\)](#)
- [Create the CloudWatch Agent Configuration File \(p. 270\)](#)
- [Metrics Collected by the CloudWatch Agent \(p. 305\)](#)
- [Common Scenarios with the CloudWatch Agent \(p. 313\)](#)
- [Troubleshooting the CloudWatch Agent \(p. 320\)](#)

Installing the CloudWatch Agent

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

Contents

- [Installing the CloudWatch Agent Using the Command Line \(p. 237\)](#)
- [Installing the CloudWatch Agent Using AWS Systems Manager \(p. 248\)](#)
- [Installing the CloudWatch Agent Using AWS CloudFormation \(p. 261\)](#)
- [Verifying the Signature of the CloudWatch Agent Package \(p. 265\)](#)

Installing the CloudWatch Agent Using the Command Line

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

Topics

- [Download and Configure the CloudWatch Agent Using the Command Line \(p. 238\)](#)
- [Create IAM Roles and Users for Use With CloudWatch Agent \(p. 242\)](#)
- [Installing and Running the CloudWatch Agent on Your Servers \(p. 243\)](#)

Download and Configure the CloudWatch Agent Using the Command Line

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

Download the CloudWatch Agent Package Using an S3 Download Link

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

For each download link, there is a general link as well as links for each Region. For example, for Amazon Linux and Amazon Linux 2 and the AMD64 architecture, three of the valid download links are:

- https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm
- https://s3.us-east-1.amazonaws.com/amazoncloudwatch-agent-us-east-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm
- https://s3.eu-central-1.amazonaws.com/amazoncloudwatch-agent-eu-central-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm

You can also download a README file about the latest changes to the agent, and a file that indicates the version number that is available for download. These files are in the following locations:

- https://s3.amazonaws.com/amazoncloudwatch-agent/info/latest/RELEASE_NOTES or https://s3.region.amazonaws.com/amazoncloudwatch-agent-Region/info/latest/RELEASE_NOTES
- https://s3.amazonaws.com/amazoncloudwatch-agent/info/latest/CWAGENT_VERSION or https://s3.region.amazonaws.com/amazoncloudwatch-agent-Region/info/latest/CWAGENT_VERSION

Architecture	Platform	Download Link	Signature File Link
AMD64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Centos	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig

Architecture	Platform	Download Link	Signature File Link
		latest/amazon-cloudwatch-agent.rpm	latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Debian	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
AMD64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig

Architecture	Platform	Download Link	Signature File Link
AMD64	Windows	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/windows/amd64/latest/amazon-cloudwatch-agent.msi	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig
ARM64	Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig

Architecture	Platform	Download Link	Signature File Link
ARM64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/arm64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig

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

1. Download the CloudWatch agent.

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

```
wget download-link
```

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

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

2. After you have downloaded the package, you can optionally verify the package signature. For more information, see [Verifying the Signature of the CloudWatch Agent Package \(p. 265\)](#).
3. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and enter the following:

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

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

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

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

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

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

Create and Modify the Agent Configuration File

After you have downloaded the CloudWatch agent, you must create the configuration file before you start the agent on any servers. For more information, see [Create the CloudWatch Agent Configuration File \(p. 270\)](#).

Create IAM Roles and Users for Use With CloudWatch Agent

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

Note

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

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

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

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane on the left, choose **Roles** and then **Create role**.
3. For **Choose the service that will use this role**, choose **EC2 Allows EC2 instances to call AWS services on your behalf**. Choose **Next: Permissions**.
4. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. If necessary, use the search box to find the policy.
5. Choose **Next: Review**.
6. Confirm that **CloudWatchAgentServerPolicy** appears next to **Policies**. In **Role name**, enter a name for the role, such as `CloudWatchAgentServerRole`. Optionally give it a description. Then choose **Create role**.

The role is now created.

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

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

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane on the left, choose **Users** and then **Add user**.
3. Enter the user name for the new user.
4. Select **Programmatic access** and choose **Next: Permissions**.
5. Choose **Attach existing policies directly**.
6. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. If necessary, use the search box to find the policy.
7. Choose **Next: Review**.
8. Confirm that the correct policies are listed, and choose **Create user**.
9. Next to the name of the new user, choose **Show**. Copy the access key and secret key to a file so that you can use them when installing the agent. Choose **Close**.

Installing and Running the CloudWatch Agent on Your Servers

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

Download the CloudWatch Agent Package Using an S3 Download Link

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

For each download link, there is a general link as well as links for each Region. For example, for Amazon Linux and Amazon Linux 2 and the AMD64 architecture, three of the valid download links are:

- https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm
- https://s3.us-east-1.amazonaws.com/amazoncloudwatch-agent-us-east-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm
- https://s3.eu-central-1.amazonaws.com/amazoncloudwatch-agent-eu-central-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm

Architecture	Platform	Download Link	Signature File Link
AMD64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Centos	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/

Architecture	Platform	Download Link	Signature File Link
		amazoncloudwatch-agent- <i>region</i> /redhat/amd64/latest/amazon-cloudwatch-agent.rpm	amazoncloudwatch-agent- <i>region</i> /redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/suse/amd64/latest/amazon-cloudwatch-agent.rpm">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Debian	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/debian/amd64/latest/amazon-cloudwatch-agent.deb">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/debian/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
AMD64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig
AMD64	Windows	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig

Architecture	Platform	Download Link	Signature File Link
ARM64	Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig
ARM64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/arm64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig

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

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

```
wget download-link
```

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

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

2. After you have downloaded the package, you can optionally verify the package signature. For more information, see [Verifying the Signature of the CloudWatch Agent Package \(p. 265\)](#).
3. Install the package. If you downloaded an RPM package on a Linux server, change to the directory containing the package and enter the following:

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

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

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

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

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

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

(Installing on an EC2 Instance) Attaching an IAM Role

To enable the CloudWatch agent to send data from the instance, you must attach an IAM role to the instance. The role to attach is **CloudWatchAgentServerRole**.

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

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

To enable the CloudWatch agent to send data from an on-premises server, you must specify the access key and secret key of the IAM user that you created earlier. For more information about creating this user, see [Create IAM Roles and Users for Use With CloudWatch Agent \(p. 242\)](#).

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

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

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

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

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

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

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

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

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

The CloudWatch agent includes a configuration file called `common-config.toml`. You can optionally use this file to specify proxy and Region information.

On a server running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent` directory.

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

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

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

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

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

- `shared_credential_profile` – For on-premises servers, this line specifies the IAM user credential profile to use to send data to CloudWatch. If you keep this line commented out, `AmazonCloudWatchAgent` is used. For more information about creating this profile, see [Installing on an On-Premises Server\) Specify IAM Credentials and AWS Region \(p. 246\)](#).

On an EC2 instance, you can use this line to have the CloudWatch agent send data from this instance to CloudWatch in a different AWS Region. To do so, specify a named profile that includes a `region` field specifying the name of the Region to send to.

If you specify a `shared_credential_profile`, you must also remove the `#` from the beginning of the `[credentials]` line.

- `shared_credential_file` – To have the agent look for credentials in a file located in a path other than the default path, specify that complete path and file name here. The default path is `/root/.aws` on Linux and is `C:\Users\Administrator\.aws` on Windows Server.

The first example below shows the syntax of a valid `shared_credential_file` line for Linux servers, and the second example is valid for Windows Server. On Windows Server, you must escape the `\` characters.

```
shared_credential_file= "/usr/username/credentials"
```

```
shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
```

If you specify a `shared_credential_file`, you must also remove the `#` from the beginning of the `[credentials]` line.

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

Start the CloudWatch Agent Using the Command Line

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

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

1. Copy the agent configuration file that you want to use to the server where you're going to run the agent. Note the pathname where you copy it to.
2. In this command, `-a fetch-config` causes the agent to load the latest version of the CloudWatch agent configuration file, and `-s` starts the agent.

On an EC2 instance running Linux, enter the following:

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

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

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

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

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -c file:configuration-file-path -s
```

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

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m onPremise -c file:configuration-file-path -s
```

Installing the CloudWatch Agent Using AWS Systems Manager

Use the following topics to install and run the CloudWatch agent using AWS Systems Manager.

Topics

- [Create IAM Roles and Users for Use with the CloudWatch Agent \(p. 249\)](#)
- [Download and Configure the CloudWatch Agent \(p. 252\)](#)
- [Installing the CloudWatch Agent on EC2 Instances Using Your Agent Configuration \(p. 253\)](#)
- [Installing the CloudWatch Agent on On-Premises Servers \(p. 257\)](#)

Create IAM Roles and Users for Use with the CloudWatch Agent

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

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

The ability to write to Parameter Store is a broad and powerful permission. You should use it only when you need it, and it shouldn't be attached to multiple instances in your deployment. If you store your CloudWatch agent configuration in Parameter Store, we recommend the following:

- Set up one instance where you perform this configuration.
- Use the IAM role with permissions to write to Parameter Store only on this instance.
- Use the IAM role with permissions to write to Parameter Store only while you are working with and saving the CloudWatch agent configuration file.

Note

We recently modified the following procedures by using new `CloudWatchAgentServerPolicy` and `CloudWatchAgentAdminPolicy` policies created by Amazon, instead of requiring customers to create these policies themselves. To use these policies to write the agent configuration file to Parameter Store and then download it from Parameter Store, your agent configuration file must have a name that starts with `AmazonCloudWatch-`. If you have a CloudWatch agent configuration file with a file name that doesn't start with `AmazonCloudWatch-`, these policies can't be used to write the file to Parameter Store or to download the file from Parameter Store.

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

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

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

Note

Parameter Store supports parameters in Standard and Advanced tiers. These parameter tiers are not related to the Basic, Standard, and Advanced levels of details available with the CloudWatch Agent predefined metric sets.

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

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles** and then choose **Create role**.
3. Under **Select type of trusted entity**, choose **AWS service**.
4. Immediately under **Choose the service that will use this role**, choose **EC2**, and then choose **Next: Permissions**.
5. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. If necessary, use the search box to find the policy.
6. To use Systems Manager to install or configure the CloudWatch agent, select the box next to **AmazonSSMManagedInstanceCore**. This AWS managed policy enables an instance to use Systems Manager service core functionality. If necessary, use the search box to find the policy. This policy isn't necessary if you start and configure the agent only through the command line.
7. Choose **Next: Tags**.
8. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose **Next: Review**.
9. For **Role name**, enter a name for your new role, such as **CloudWatchAgentServerRole** or another name that you prefer.
10. (Optional) For **Role description**, enter a description.
11. Confirm that **CloudWatchAgentServerPolicy** and optionally **AmazonSSMManagedInstanceCore** appear next to **Policies**.
12. Choose **Create role**.

The role is now created.

The following procedure creates the IAM role that can also write to Parameter Store. You can use this role to store the agent configuration file in Parameter Store so that other servers can retrieve it.

The permissions for writing to Parameter Store provide broad access. This role shouldn't be attached to all your servers, and only administrators should use it. After you create the agent configuration file and copy it to Parameter Store, you should detach this role from the instance and use **CloudWatchAgentServerRole** instead.

To create the IAM role for an administrator to write to Parameter Store

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles** and then choose **Create role**.
3. Under **Select type of trusted entity**, choose **AWS service**.
4. Immediately under **Choose the service that will use this role**, choose **EC2**, and then choose **Next: Permissions**.
5. In the list of policies, select the check box next to **CloudWatchAgentAdminPolicy**. If necessary, use the search box to find the policy.
6. To use Systems Manager to install or configure the CloudWatch agent, select the box next to **AmazonSSMManagedInstanceCore**. This AWS managed policy enables an instance to use Systems Manager service core functionality. If necessary, use the search box to find the policy. This policy isn't necessary if you start and configure the agent only through the command line.
7. Choose **Next: Tags**.
8. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose **Next: Review**.

9. For **Role name**, enter a name for your new role, such as **CloudWatchAgentAdminRole** or another name that you prefer.
10. (Optional) For **Role description**, enter a description.
11. Confirm that **CloudWatchAgentAdminPolicy** and optionally **AmazonSSMManagedInstanceCore** appear next to **Policies**.
12. Choose **Create role**.

The role is now created.

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

The first procedure creates the IAM user that you need to run the CloudWatch agent. This user provides permissions to send data to CloudWatch.

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

Note

Parameter Store supports parameters in Standard and Advanced tiers. These parameter tiers are not related to the Basic, Standard, and Advanced levels of details available with the CloudWatch Agent predefined metric sets.

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

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**, and then choose **Add user**.
3. Enter the user name for the new user.
4. For **Access type**, select **Programmatic access**, and then choose **Next: Permissions**.
5. For **Set permissions**, choose **Attach existing policies directly**.
6. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. If necessary, use the search box to find the policy.
7. To use Systems Manager to install or configure the CloudWatch agent, select the box next to **AmazonSSMManagedInstanceCore**. This AWS managed policy enables an instance to use Systems Manager service core functionality. (If necessary, use the search box to find the policy. This policy isn't necessary if you start and configure the agent only through the command line.)
8. Choose **Next: Tags**.
9. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose **Next: Review**.
10. Confirm that the correct policies are listed, and then choose **Create user**.
11. In the row for the new user, choose **Show**. Copy the access key and secret key to a file so that you can use them when installing the agent. Choose **Close**.

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

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

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**, and then choose **Add user**.
3. Enter the user name for the new user.
4. For **Access type**, select **Programmatic access**, and then choose **Next: Permissions**.
5. For **Set permissions**, choose **Attach existing policies directly**.
6. In the list of policies, select the check box next to **CloudWatchAgentAdminPolicy**. If necessary, use the search box to find the policy.
7. To use Systems Manager to install or configure the CloudWatch agent, select the check box next to **AmazonSSMManagedInstanceCore**. This AWS managed policy enables an instance to use Systems Manager service core functionality. (If necessary, use the search box to find the policy. This policy isn't necessary if you start and configure the agent only through the command line.)
8. Choose **Next: Tags**.
9. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose **Next: Review**.
10. Confirm that the correct policies are listed, and then choose **Create user**.
11. In the row for the new user, choose **Show**. Copy the access key and secret key to a file so that you can use them when installing the agent. Choose **Close**.

Download and Configure the CloudWatch Agent

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

Installing or Updating SSM Agent

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

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

For information about installing or updating the SSM Agent, see [Working with SSM Agent](#) in the *AWS Systems Manager User Guide*.

(Optional) Verify Systems Manager Prerequisites

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

Verify Internet Access

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

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

- If you're using the agent to collect metrics, you must whitelist the CloudWatch endpoints for the appropriate Regions. These endpoints are listed in [Amazon CloudWatch](#) in the *Amazon Web Services General Reference*.
- If you're using the agent to collect logs, you must whitelist the CloudWatch Logs endpoints for the appropriate Regions. These endpoints are listed in [Amazon CloudWatch Logs](#) in the *Amazon Web Services General Reference*.
- If you're using Systems Manager to install the agent or Parameter Store to store your configuration file, you must whitelist the Systems Manager endpoints for the appropriate Regions. These endpoints are listed in [AWS Systems Manager](#) in the *Amazon Web Services General Reference*.

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

To download the CloudWatch agent using Systems Manager

1. Open the Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
 2. In the navigation pane, choose **Run Command**.
- or-
- If the AWS Systems Manager home page opens, scroll down and choose **Explore Run Command**.
3. Choose **Run command**.
 4. In the **Command document** list, choose **AWS-ConfigureAWSPackage**.
 5. In the **Targets** area, choose the instance to install the CloudWatch agent on. If you don't see a specific instance, it might not be configured as a managed instance for use with Systems Manager. For more information, see [Setting Up AWS Systems Manager for Hybrid Environments](#) in the *AWS Systems Manager User Guide*.
 6. In the **Action** list, choose **Install**.
 7. In the **Name** field, enter **AmazonCloudWatchAgent**.
 8. Keep **Version** set to **latest** to install the latest version of the agent.
 9. Choose **Run**.
 10. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully installed.

Create and Modify the Agent Configuration File

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

If you're going to save your agent configuration file in the Systems Manager Parameter Store, you must use an EC2 instance to save to the Parameter Store. Additionally, you must first attach to that instance the `CloudWatchAgentAdminRole` IAM role. For more information about attaching roles, see [Attaching an IAM Role to an Instance](#) in the *Amazon EC2 User Guide for Windows Instances*.

For more information about creating the CloudWatch agent configuration file, see [Create the CloudWatch Agent Configuration File](#) (p. 270).

Installing the CloudWatch Agent on EC2 Instances Using Your Agent Configuration

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

Topics

- [Attach an IAM Role to the Instance \(p. 254\)](#)
- [Download the CloudWatch Agent Package on an Amazon EC2 Instance \(p. 254\)](#)
- [\(Optional\) Modify the Common Configuration and Named Profile for CloudWatch Agent \(p. 255\)](#)
- [Start the CloudWatch Agent \(p. 256\)](#)

Attach an IAM Role to the Instance

You must attach the **CloudWatchAgentServerRole** IAM role to the EC2 instance to be able to run the CloudWatch agent on the instance. This role enables the CloudWatch agent to perform actions on the instance.

For more information, see [Attaching an IAM Role to an Instance](#) in the *Amazon EC2 User Guide for Windows Instances*.

Download the CloudWatch Agent Package on an Amazon EC2 Instance

You can download the CloudWatch agent package using either Systems Manager Run Command or an Amazon S3 download link. For information about using an Amazon S3 download link, see [Download the CloudWatch Agent Package Using an S3 Download Link \(p. 238\)](#).

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

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

Installing or Updating SSM Agent

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

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

For information about installing or updating SSM Agent on an instance running Windows Server, see [Installing and Configuring SSM Agent on Windows Instances](#) in the *AWS Systems Manager User Guide*.

(Optional) Verify Systems Manager Prerequisites

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

Verify Internet Access

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

Download the CloudWatch Agent Package

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

To download the CloudWatch agent using Run Command

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

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

-or-

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

3. Choose **Run command**.
4. In the **Command document** list, choose **AWS-ConfigureAWSPackage**.
5. In the **Targets** area, choose the instance on which to install the CloudWatch agent. If you do not see a specific instance, it might not be configured for Run Command. For more information, see [Setting Up AWS Systems Manager for Hybrid Environments](#) in the *AWS Systems Manager User Guide*.
6. In the **Action** list, choose **Install**.
7. In the **Name** box, enter **AmazonCloudWatchAgent**.
8. Keep **Version** set to **latest** to install the latest version of the agent.
9. Choose **Run**.
10. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully installed.

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

The CloudWatch agent includes a configuration file called `common-config.toml`. You can use this file optionally specify proxy and Region information.

On a server running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent` directory.

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

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

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

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

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

- `shared_credential_profile` – For on-premises servers, this line specifies the IAM user credential profile to use to send data to CloudWatch. If you keep this line commented out, `AmazonCloudWatchAgent` is used.

On an EC2 instance, you can use this line to have the CloudWatch agent send data from this instance to CloudWatch in a different AWS Region. To do so, specify a named profile that includes a `region` field specifying the name of the Region to send to.

If you specify a `shared_credential_profile`, you must also remove the `#` from the beginning of the `[credentials]` line.

- `shared_credential_file` – To have the agent look for credentials in a file located in a path other than the default path, specify that complete path and file name here. The default path is `/root/.aws` on Linux and is `C:\\Users\\Administrator\\.aws` on Windows Server.

The first example below shows the syntax of a valid `shared_credential_file` line for Linux servers, and the second example is valid for Windows Server. On Windows Server, you must escape the `\` characters.

```
shared_credential_file= "/usr/username/credentials"
```

```
shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\.credentials"
```

If you specify a `shared_credential_file`, you must also remove the `#` from the beginning of the `[credentials]` line.

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

Start the CloudWatch Agent

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

Start the CloudWatch Agent Using Systems Manager Run Command

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

To start the CloudWatch agent using Run Command

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

-or-

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

3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManageAgent**.
5. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
6. In the **Action** list, choose **configure**.
7. In the **Optional Configuration Source** list, choose **ssm**.
8. In the **Optional Configuration Location** box, enter the name of the agent configuration file that you created and saved to Systems Manager Parameter Store, as explained in [Create the CloudWatch Agent Configuration File \(p. 270\)](#).
9. In the **Optional Restart** list, choose **yes** to start the agent after you have finished these steps.
10. Choose **Run**.
11. Optionally, in the **Targets and outputs** areas, select the button next to an instance name and choose **View output**. Systems Manager should show that the agent was successfully started.

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

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

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

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

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

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

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

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

Windows Server: If you saved the agent configuration file in Systems Manager Parameter Store, enter the following from the PowerShell console:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -c ssm:configuration-parameter-store-name -s
```

Windows Server: If you saved the agent configuration file on the local computer, enter the following from the PowerShell console:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent\config.json" -s
```

Installing the CloudWatch Agent on On-Premises Servers

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

Download the CloudWatch Agent on an On-Premises Server

You can download the CloudWatch agent package using either Systems Manager Run Command or an Amazon S3 download link. For information about using an Amazon S3 download link, see [Download the CloudWatch Agent Package Using an S3 Download Link \(p. 238\)](#).

Download Using Systems Manager

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

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

For information about updating SSM Agent on a server running Linux, see [Install SSM Agent for a Hybrid Environment \(Linux\)](#) in the *AWS Systems Manager User Guide*.

For information about updating the SSM Agent on a server running Windows Server, see [Install SSM Agent for a Hybrid Environment \(Windows\)](#) in the *AWS Systems Manager User Guide*.

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

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

-or-

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

3. Choose **Run command**.
4. In the **Command document** list, select the button next to **AWS-ConfigureAWSPackage**.
5. In the **Targets** area, select the server to install the CloudWatch agent on. If you don't see a specific server, it might not be configured for Run Command. For more information, see [Setting Up AWS Systems Manager for Hybrid Environments](#) in the *AWS Systems Manager User Guide*.
6. In the **Action** list, choose **Install**.
7. In the **Name** box, enter `AmazonCloudWatchAgent`.
8. Keep **Version** blank to install the latest version of the agent.
9. Choose **Run**.

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

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

To enable the CloudWatch agent to send data from an on-premises server, you must specify the access key and secret key of the IAM user that you created earlier. For more information about creating this user, see [Create IAM Roles and Users for Use with the CloudWatch Agent](#) (p. 249).

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

Following is an example of this file.

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

For `my_access_key` and `my_secret_key`, use the keys from the IAM user that doesn't have the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for CloudWatch agent, see [Create IAM Users to Use with the CloudWatch Agent on On-Premises Servers](#) (p. 251).

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

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

To create the AmazonCloudWatchAgent profile for the CloudWatch agent

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

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

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

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

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

The CloudWatch agent includes a configuration file called `common-config.toml`. You can optionally use this file to specify proxy and Region information.

On a server running Linux, this file is in the `/opt/aws/amazon-cloudwatch-agent/etc` directory. On a server running Windows Server, this file is in the `C:\ProgramData\Amazon\AmazonCloudWatchAgent` directory.

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

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

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

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

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

- `shared_credential_profile` – For on-premises servers, this line specifies the IAM user credential profile to use to send data to CloudWatch. If you keep this line commented out, `AmazonCloudWatchAgent` is used. For more information about creating this profile, see [\(Installing on an On-Premises Server\) Specify IAM Credentials and AWS Region \(p. 258\)](#).

On an EC2 instance, you can use this line to have the CloudWatch agent send data from this instance to CloudWatch in a different AWS Region. To do so, specify a named profile that includes a `region` field specifying the name of the Region to send to.

If you specify a `shared_credential_profile`, you must also remove the `#` from the beginning of the `[credentials]` line.

- `shared_credential_file` – To have the agent look for credentials in a file located in a path other than the default path, specify that complete path and file name here. The default path is `/root/.aws` on Linux and is `C:\Users\Administrator\.aws` on Windows Server.

The first example below shows the syntax of a valid `shared_credential_file` line for Linux servers, and the second example is valid for Windows Server. On Windows Server, you must escape the `\` characters.

```
shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
```

```
shared_credential_file= "/usr/username/credentials"
```

If you specify a `shared_credential_file`, you must also remove the `#` from the beginning of the `[credentials]` line.

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

Starting the CloudWatch Agent

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

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

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

-or-

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

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

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

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

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

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

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

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

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

Windows Server: If you saved the agent configuration file in Systems Manager Parameter Store, enter the following from the PowerShell console:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a  
fetch-config -m onPremise -c ssm:configuration-parameter-store-name -s
```

Windows Server: If you saved the agent configuration file on the local computer, enter the following from the PowerShell console:

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a  
fetch-config -m onPremise -c file:configuration-file-path -s
```

Installing the CloudWatch Agent Using AWS CloudFormation

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

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

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

To modify these templates to collect different metrics, modify the following section of the template. The following example is from the template for Linux servers. Follow the format and syntax of the agent configuration file to make these changes. For more information, see [Manually Create or Edit the CloudWatch Agent Configuration File \(p. 275\)](#).

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

Note

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

- The templates in the `ssm` directory load an agent configuration file from Parameter Store. To use these templates, you must first create a configuration file and upload it to Parameter Store. You then provide the Parameter Store name of the file in the template. You can create the configuration file manually or by using the wizard. For more information, see [Create the CloudWatch Agent Configuration File](#) (p. 270).

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

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

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

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

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

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

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

2. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
3. Choose **Create stack**.
4. For **Choose a template**, select **Upload a template to Amazon S3**, choose the downloaded template, and choose **Next**.
5. On the **Specify Details** page, fill out the following parameters and choose **Next**:
 - **Stack name**: Choose a stack name for your AWS CloudFormation stack.
 - **IAMRole**: Choose an IAM role that has permissions to write CloudWatch metrics and logs. For more information, see [Create IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances](#) (p. 242).
 - **InstanceAMI**: Choose an AMI that is valid in the Region where you're going to launch your stack.
 - **InstanceType**: Choose a valid instance type.
 - **KeyName**: To enable SSH access to the new instance, choose an existing Amazon EC2 key pair. If you don't already have an Amazon EC2 key pair, you can create one in the AWS Management Console. For more information, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.
 - **SSHLocation**: Specifies the IP address range that can be used to connect to the instance using SSH. The default allows access from any IP address.
6. On the **Options** page, you can choose to tag your stack resources. Choose **Next**.
7. On the **Review** page, review your information, acknowledge that the stack might create IAM resources, and then choose **Create**.

If you refresh the console, you see that the new stack has the `CREATE_IN_PROGRESS` status.

8. When the instance is created, you can see it in the Amazon EC2 console. Optionally, you can connect to the host and check the progress.

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

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

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

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

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

To use AWS CloudFormation to update the CloudWatch agent

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

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

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

After some time, you see `UPDATE_COMPLETE`.

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

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

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

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

1. If you haven't done so already, download the CloudWatch agent package to one of your computers so that you can create the agent configuration file. For more information and downloading the agent using Parameter Store, see [Download and Configure the CloudWatch Agent \(p. 252\)](#).

For more information on downloading the package using the command line, see [Download and Configure the CloudWatch Agent Using the Command Line](#) (p. 238).

2. Create the agent configuration file and save it in Parameter Store. For more information, see [Create the CloudWatch Agent Configuration File](#) (p. 270).
3. Download the template from GitHub as follows:

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

4. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
5. Choose **Create stack**.
6. For **Choose a template**, select **Upload a template to Amazon S3**, choose the template that you downloaded, and choose **Next**.
7. On the **Specify Details** page, fill out the following parameters accordingly and choose **Next**:
 - **Stack name**: Choose a stack name for your AWS CloudFormation stack.
 - **IAMRole**: Choose an IAM role that has permissions to write CloudWatch metrics and logs. For more information, see [Create IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances](#) (p. 249).
 - **InstanceAMI**: Choose an AMI that is valid in the Region where you're going to launch your stack.
 - **InstanceType**: Choose a valid instance type.
 - **KeyName**: To enable SSH access to the new instance, choose an existing Amazon EC2 key pair. If you don't already have an Amazon EC2 key pair, you can create one in the AWS Management Console. For more information, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.
 - **SSHLocation**: Specifies the IP address range that can be used to connect to the instance using SSH. The default allows access from any IP address.
 - **SSMKey**: Specifies the agent configuration file that you created and saved in Parameter Store.
8. On the **Options** page, you can choose to tag your stack resources. Choose **Next**.
9. On the **Review** page, review your information, acknowledge that the stack might create IAM resources, and then choose **Create**.

If you refresh the console, you see that the new stack has the `CREATE_IN_PROGRESS` status.

10. When the instance is created, you can see it in the Amazon EC2 console. Optionally, you can connect to the host and check the progress.

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

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

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

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

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

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

1. Change the agent configuration file stored in Parameter Store to the new configuration that you want.

2. In the AWS CloudFormation template that you downloaded in the [the section called “Tutorial: Install the CloudWatch Agent Using AWS CloudFormation and Parameter Store”](#) (p. 263) topic, change the version number. For example, you might change `VERSION=1.0` to `VERSION=2.0`.
3. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
4. On the AWS CloudFormation dashboard, select the stack that you created and choose **Update Stack**.
5. For **Select Template**, select **Upload a template to Amazon S3**, select the template that you just modified, and choose **Next**.
6. On the **Options** page, choose **Next** and then **Next**.
7. On the **Review** page, review your information and choose **Update**.

After some time, you see `UPDATE_COMPLETE`.

Troubleshooting Installation of the CloudWatch Agent with AWS CloudFormation

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

Detecting When an Update Fails

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

Metrics Are Missing

If you don't see metrics that you expect to see after installing or updating the agent, confirm that the agent is configured to collect that metric. To do this, check the `amazon-cloudwatch-agent.json` file to make sure that the metric is listed, and check that you are looking in the correct metric namespace. For more information, see [CloudWatch Agent Files and Locations](#) (p. 323).

Verifying the Signature of the CloudWatch Agent Package

GPG signature files are included for CloudWatch agent packages on Linux servers. You can use a public key to verify that the agent download file is original and unmodified.

For Windows Server, you can use the MSI to verify the signature.

To find the correct signature file, see the following table. For each architecture and operating system there is a general link as well as links for each Region. For example, for Amazon Linux and Amazon Linux 2 and the AMD64 architecture, three of the valid links are:

- https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
- https://s3.us-east-1.amazonaws.com/amazoncloudwatch-agent-us-east-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
- https://s3.eu-central-1.amazonaws.com/amazoncloudwatch-agent-eu-central-1/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig

Architecture	Platform	Download Link	Signature File Link
AMD64	Amazon Linux and Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Centos	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/centos/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/centos/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/redhat/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/amd64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/amd64/latest/amazon-cloudwatch-agent.rpm.sig
AMD64	Debian	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb https://s3.region.amazonaws.com/	https://s3.amazonaws.com/amazoncloudwatch-agent/debian/amd64/latest/amazon-cloudwatch-agent.deb.sig https://s3.region.amazonaws.com/

Architecture	Platform	Download Link	Signature File Link
		amazoncloudwatch-agent- <i>region</i> /debian/amd64/latest/amazon-cloudwatch-agent.deb	amazoncloudwatch-agent- <i>region</i> /debian/amd64/latest/amazon-cloudwatch-agent.deb.sig
AMD64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb.sig
AMD64	Windows	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi	https://s3.amazonaws.com/amazoncloudwatch-agent/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/windows/amd64/latest/amazon-cloudwatch-agent.msi.sig
ARM64	Amazon Linux 2	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/amazon_linux/arm64/latest/amazon-cloudwatch-agent.rpm.sig
ARM64	Redhat	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/redhat/arm64/latest/amazon-cloudwatch-agent.rpm">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/redhat/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig <a href="https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig">https://s3.<i>region</i>.amazonaws.com/amazoncloudwatch-agent-<i>region</i>/redhat/arm64/latest/amazon-cloudwatch-agent.rpm.sig

Architecture	Platform	Download Link	Signature File Link
ARM64	Ubuntu	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig
		https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb	https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/ubuntu/arm64/latest/amazon-cloudwatch-agent.deb.sig
ARM64	SUSE	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.amazonaws.com/amazoncloudwatch-agent/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig
		https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/arm64/latest/amazon-cloudwatch-agent.rpm	https://s3.region.amazonaws.com/amazoncloudwatch-agent-region/suse/arm64/latest/amazon-cloudwatch-agent.rpm.sig

To verify the CloudWatch agent package on a Linux server

1. Download the public key.

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

2. Import the public key into your keyring.

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

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

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

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

The fingerprint string should be equal to the following:

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

If the fingerprint string doesn't match, don't install the agent. Contact Amazon Web Services.

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

4. Download the package signature file using **wget**. To determine the correct signature file, see the preceding table.

```
wget Signature File Link
```

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

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

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

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

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

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

You can perform the remaining steps in Windows PowerShell.

2. Download the public key.

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

3. Import the public key into your keyring.

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

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

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

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

The fingerprint string should be equal to the following:

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

If the fingerprint string doesn't match, don't install the agent. Contact Amazon Web Services.

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

5. Download the package signature file using `wget`. To determine the correct signature file, see [CloudWatch Agent Download Links \(p. 238\)](#).
6. To verify the signature, run `gpg --verify`.

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

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

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

Create the CloudWatch Agent Configuration File

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

The agent configuration file is a JSON file that specifies the metrics and logs that the agent is to collect, including custom metrics. You can create it by using the wizard or by creating it yourself from scratch. You could also use the wizard to initially create the configuration file and then modify it manually. If you create or modify the file manually, the process is more complex, but you have more control over the metrics collected and can specify metrics not available through the wizard.

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

After you have created a configuration file, you can save it manually as a JSON file and then use this file when installing the agent on your servers. Alternatively, you can store it in Systems Manager Parameter Store if you're going to use Systems Manager when you install the agent on servers.

Contents

- [Create the CloudWatch Agent Configuration File with the Wizard \(p. 270\)](#)
- [Manually Create or Edit the CloudWatch Agent Configuration File \(p. 275\)](#)

Create the CloudWatch Agent Configuration File with the Wizard

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

- Are you installing the agent on an Amazon EC2 instance or an on-premises server?
- Is the server running Linux or Windows Server?

- Do you want the agent to also send log files to CloudWatch Logs? If so, do you have an existing CloudWatch Logs agent configuration file? If yes, the CloudWatch agent can use this file to determine the logs to collect from the server.
- If you're going to collect metrics from the server, do you want to monitor one of the default sets of metrics or customize the list of metrics that you collect?
- Do you want to collect custom metrics from your applications or services, using StatsD or collectd?
- Are you migrating from an existing SSM Agent?

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

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

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

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

For `my_access_key` and `my_secret_key`, use the keys from the IAM user that has the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for the CloudWatch agent, see [Create IAM Users to Use with the CloudWatch Agent on On-Premises Servers](#) (p. 251).

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

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

CloudWatch Agent Predefined Metric Sets

The wizard is configured with predefined sets of metrics, with different detail levels. These sets of metrics are shown in the following tables. For more information about these metrics, see [Metrics Collected by the CloudWatch Agent](#) (p. 305).

Note

Parameter Store supports parameters in Standard and Advanced tiers. These parameter tiers are not related to the Basic, Standard, and Advanced levels of metric details that are described in these tables.

Amazon EC2 instances running Linux

Detail Level	Metrics Included
Basic	<p>Mem: mem_used_percent</p> <p>Disk: disk_used_percent</p> <p>The disk metrics such as <code>disk_used_percent</code> have a dimension for <code>Partition</code>, which means that the number of custom metrics generated is dependent on the number of partitions associated with your instance. The</p>

Detail Level	Metrics Included
	number of disk partitions you have depends on which AMI you are using and the number of Amazon EBS volumes you attach to the server.
Standard	CPU: <code>cpu_usage_idle</code> , <code>cpu_usage_iowait</code> , <code>cpu_usage_user</code> , <code>cpu_usage_system</code> Disk: <code>disk_used_percent</code> , <code>disk_inodes_free</code> Diskio: <code>diskio_io_time</code> Mem: <code>mem_used_percent</code> Swap: <code>swap_used_percent</code>
Advanced	CPU: <code>cpu_usage_idle</code> , <code>cpu_usage_iowait</code> , <code>cpu_usage_user</code> , <code>cpu_usage_system</code> Disk: <code>disk_used_percent</code> , <code>disk_inodes_free</code> Diskio: <code>diskio_io_time</code> , <code>diskio_write_bytes</code> , <code>diskio_read_bytes</code> , <code>diskio_writes</code> , <code>diskio_reads</code> Mem: <code>mem_used_percent</code> Netstat: <code>netstat_tcp_established</code> , <code>netstat_tcp_time_wait</code> Swap: <code>swap_used_percent</code>

On-premises servers running Linux

Detail Level	Metrics Included
Basic	Disk: <code>disk_used_percent</code> Diskio: <code>diskio_write_bytes</code> , <code>diskio_read_bytes</code> , <code>diskio_writes</code> , <code>diskio_reads</code> Mem: <code>mem_used_percent</code> Net: <code>net_bytes_sent</code> , <code>net_bytes_recv</code> , <code>net_packets_sent</code> , <code>net_packets_recv</code> Swap: <code>swap_used_percent</code>
Standard	CPU: <code>cpu_usage_idle</code> , <code>cpu_usage_iowait</code> Disk: <code>disk_used_percent</code> , <code>disk_inodes_free</code> Diskio: <code>diskio_io_time</code> , <code>diskio_write_bytes</code> , <code>diskio_read_bytes</code> , <code>diskio_writes</code> , <code>diskio_reads</code> Mem: <code>mem_used_percent</code> Net: <code>net_bytes_sent</code> , <code>net_bytes_recv</code> , <code>net_packets_sent</code> , <code>net_packets_recv</code> Swap: <code>swap_used_percent</code>

Detail Level	Metrics Included
Advanced	<p>CPU: cpu_usage_guest, cpu_usage_idle, cpu_usage_iowait, cpu_usage_steal, cpu_usage_user, cpu_usage_system</p> <p>Disk: disk_used_percent, disk_inodes_free</p> <p>Diskio: diskio_io_time, diskio_write_bytes, diskio_read_bytes, diskio_writes, diskio_reads</p> <p>Mem: mem_used_percent</p> <p>Net: net_bytes_sent, net_bytes_recv, net_packets_sent, net_packets_recv</p> <p>Netstat: netstat_tcp_established, netstat_tcp_time_wait</p> <p>Swap: swap_used_percent</p>

Amazon EC2 instances running Windows Server

Detail Level	Metrics Included
Basic	<p>Memory: Memory % Committed Bytes In Use</p> <p>LogicalDisk: LogicalDisk % Free Space</p>
Standard	<p>Memory: Memory % Committed Bytes In Use</p> <p>Paging: Paging File % Usage</p> <p>Processor: Processor % Idle Time, Processor % Interrupt Time, Processor % User Time</p> <p>PhysicalDisk: PhysicalDisk % Disk Time</p> <p>LogicalDisk: LogicalDisk % Free Space</p>
Advanced	<p>Memory: Memory % Committed Bytes In Use</p> <p>Paging: Paging File % Usage</p> <p>Processor: Processor % Idle Time, Processor % Interrupt Time, Processor % User Time</p> <p>LogicalDisk: LogicalDisk % Free Space</p> <p>PhysicalDisk: PhysicalDisk % Disk Time, PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec</p> <p>TCP: TCPv4 Connections Established, TCPv6 Connections Established</p>

On-premises server running Windows Server

Detail Level	Metrics Included
Basic	<p>Paging: Paging File % Usage</p> <p>Processor: Processor % Processor Time</p> <p>LogicalDisk: LogicalDisk % Free Space</p> <p>PhysicalDisk: PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec</p> <p>Memory: Memory % Committed Bytes In Use</p> <p>Network Interface: Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec, Network Interface Packets Sent/sec, Network Interface Packets Received/sec</p>
Standard	<p>Paging: Paging File % Usage</p> <p>Processor: Processor % Processor Time, Processor % Idle Time, Processor % Interrupt Time</p> <p>LogicalDisk: LogicalDisk % Free Space</p> <p>PhysicalDisk: PhysicalDisk % Disk Time, PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec</p> <p>Memory: Memory % Committed Bytes In Use</p> <p>Network Interface: Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec, Network Interface Packets Sent/sec, Network Interface Packets Received/sec</p>
Advanced	<p>Paging: Paging File % Usage</p> <p>Processor: Processor % Processor Time, Processor % Idle Time, Processor % Interrupt Time, Processor % User Time</p> <p>LogicalDisk: LogicalDisk % Free Space</p> <p>PhysicalDisk: PhysicalDisk % Disk Time, PhysicalDisk Disk Write Bytes/sec, PhysicalDisk Disk Read Bytes/sec, PhysicalDisk Disk Writes/sec, PhysicalDisk Disk Reads/sec</p> <p>Memory: Memory % Committed Bytes In Use</p> <p>Network Interface: Network Interface Bytes Sent/sec, Network Interface Bytes Received/sec, Network Interface Packets Sent/sec, Network Interface Packets Received/sec</p> <p>TCP: TCPv4 Connections Established, TCPv6 Connections Established</p>

Run the CloudWatch Agent Configuration Wizard

To create the CloudWatch agent configuration file

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

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

On a server running Windows Server, enter the following:

```
cd "C:\Program Files\Amazon\AmazonCloudWatchAgent"  
amazon-cloudwatch-agent-config-wizard.exe
```

2. Answer the questions to customize the configuration file for your server.
3. If you're storing the configuration file locally, the configuration file `config.json` is stored in `/opt/aws/amazon-cloudwatch-agent/bin/` on Linux servers, and is stored in `C:\Program Files\Amazon\AmazonCloudWatchAgent` on Windows Server. You can then copy this file to other servers where you want to install the agent.

If you're going to use Systems Manager to install and configure the agent, be sure to answer **Yes** when prompted whether to store the file in Systems Manager Parameter Store. You can also choose to store the file in Parameter Store even if you aren't using the SSM Agent to install the CloudWatch agent. To be able to store the file in Parameter Store, you must use an IAM role with sufficient permissions. For more information, see [Create IAM Roles and Users for Use with the CloudWatch Agent](#) (p. 249).

Manually Create or Edit the CloudWatch Agent Configuration File

The CloudWatch agent configuration file is a JSON file with three sections: `agent`, `metrics`, and `logs`.

- The `agent` section includes fields for the overall configuration of the agent. If you use the wizard, it doesn't create an `agent` section.
- The `metrics` section specifies the custom metrics for collection and publishing to CloudWatch. If you're using the agent only to collect logs, you can omit the `metrics` section from the file.
- The `logs` section specifies what log files are published to CloudWatch Logs. This can include events from the Windows Event Log if the server runs Windows Server.

The following sections explain the structure and fields of this JSON file. You can also view the schema definition for this configuration file. The schema definition is located at `installation-directory/doc/amazon-cloudwatch-agent-schema.json` on Linux servers, and at `installation-directory/amazon-cloudwatch-agent-schema.json` on servers running Windows Server.

If you create or edit the agent configuration file manually, you can give it any name. For simplicity in troubleshooting, we recommend that you name it `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` on a Linux server and `%Env:ProgramData%\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json` on servers running Windows Server. After you have created the file, you can copy it to other servers where you want to install the agent.

CloudWatch Agent Configuration File: Agent Section

The `agent` section can include the following fields. The wizard doesn't create an `agent` section. Instead, the wizard omits it and uses the default values for all fields in this section.

- `metrics_collection_interval` – Optional. Specifies how often all metrics specified in this configuration file are to be collected. You can override this value for specific types of metrics.

This value is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

The default value is 60.

- `region` – Specifies the Region to use for the CloudWatch endpoint when an Amazon EC2 instance is being monitored. The metrics collected are sent to this Region, such as `us-west-1`. If you omit this field, the agent sends metrics to the Region where the Amazon EC2 instance is located.

If you are monitoring an on-premises server, this field isn't used, and the agent reads the Region from the `AmazonCloudWatchAgent` profile of the AWS configuration file.

- `credentials` – Specifies an IAM role to use when sending metrics and logs to a different AWS account. If specified, this field contains one parameter, `role_arn`.
 - `role_arn` – Specifies the Amazon Resource Name (ARN) of an IAM role to use for authentication when sending metrics and logs to a different AWS account. For more information, see [Sending Metrics and Logs to a Different Account \(p. 318\)](#).
- `debug` – Optional. Specifies running the CloudWatch agent with debug log messages. The default value is `false`.
- `logfile` – Specifies the location where the CloudWatch agent writes log messages. If you specify an empty string, the log goes to `stderr`. If you don't specify this option, the default locations are the following:
 - Linux: `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`
 - Windows Server: `c:\ProgramData\Amazon\CloudWatchAgent\Logs\amazon-cloudwatch-agent.log`

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

- `omit_hostname` – Optional. By default, the hostname is published as a dimension of metrics that are collected by the agent. Set this value to `true` to prevent the hostname from being published as a dimension. The default value is `false`.
- `run_as_user` – Optional. Specifies a user to use to run the CloudWatch agent. If you don't specify this parameter, the root user is used. This option is valid only on Linux servers.

If you specify this option, the user must exist before you start the CloudWatch agent. For more information, see [Running the CloudWatch Agent as a Different User \(p. 313\)](#).

The following is an example of an agent section.

```
"agent": {
  "metrics_collection_interval": 60,
  "region": "us-west-1",
  "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",
  "debug": false,
  "run_as_user": "cwagent"
}
```

CloudWatch Agent Configuration File: Metrics Section

On servers running either Linux or Windows Server, the `metrics` section includes the following fields:

- `namespace` – Optional. The namespace to use for the metrics collected by the agent. The default value is `CWAgent`. The maximum length is 255 characters.
- `append_dimensions` – Optional. Adds Amazon EC2 metric dimensions to all metrics collected by the agent. The only supported key-value pairs are shown in the following list. Any other key-value pairs are ignored.
 - `"ImageID": "${aws:ImageID}"` sets the instance's AMI ID as the value of the `ImageID` dimension.
 - `"InstanceId": "${aws:InstanceId}"` sets the instance's instance ID as the value of the `InstanceId` dimension.
 - `"InstanceType": "${aws:InstanceType}"` sets the instance's instance type as the value of the `InstanceType` dimension.
 - `"AutoScalingGroupName": "${aws:AutoScalingGroupName}"` sets the instance's Auto Scaling group name as the value of the `AutoScalingGroupName` dimension.

If you want to append dimensions to metrics with arbitrary key-value pairs, use the `append_dimensions` parameter in the field for that particular type of metric.

If you specify a value that depends on Amazon EC2 metadata and you use proxies, you must make sure that the server can access the endpoint for Amazon EC2. For more information about these endpoints, see [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) in the *Amazon Web Services General Reference*.

- `aggregation_dimensions` – Optional. Specifies the dimensions that collected metrics are to be aggregated on. For example, if you roll up metrics on the `AutoScalingGroupName` dimension, the metrics from all instances in each Auto Scaling group are aggregated and can be viewed as a whole.

You can roll up metrics along single or multiple dimensions. For example, specifying `[["InstanceId"], ["InstanceType"], ["InstanceId", "InstanceType"]]` aggregates metrics for instance ID singly, instance type singly, and for the combination of the two dimensions.

You can also specify `[]` to roll up all metrics into one collection, disregarding all dimensions.

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

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

For example, the following part of the metrics section of the configuration file sets the agent to use a VPC Endpoint when sending metrics.

```
{
  "metrics": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXXXX.monitoring.us-
east-1.vpce.amazonaws.com",
    .....
  },
}
```

- `metrics_collected` – Required. Specifies which metrics are to be collected, including custom metrics collected through `StatsD` or `collectd`. This section includes several subsections.

The contents of the `metrics_collected` section depend on whether this configuration file is for a server running Linux or Windows Server.

- `force_flush_interval` – Specifies in seconds the maximum amount of time that metrics remain in the memory buffer before being sent to the server. No matter the setting for this, if the size of the

metrics in the buffer reaches 40 KB or 20 different metrics, the metrics are immediately sent to the server.

The default value is 60.

- `credentials` – Specifies an IAM role to use when sending metrics to a different account. If specified, this field contains one parameter, `role_arn`.
- `role_arn` – Specifies the ARN of an IAM role to use for authentication when sending metrics to a different account. For more information, see [Sending Metrics and Logs to a Different Account \(p. 318\)](#). If specified here, this value overrides the `role_arn` specified in the `agent` section of the configuration file, if any.

Linux

On servers running Linux, the `metrics_collected` section of the configuration file can also contain the following fields.

Many of these fields can include a `measurement` sections that lists the metrics you want to collect for that resource. These `measurement` sections can either specify the complete metric name such as `swap_used`, or just the part of the metric name that will be appended to the type of resource. For example, specifying `reads` in the `measurement` section of the `diskio` section causes the `diskio_reads` metric to be collected.

- `collectd` – Optional. Specifies that you want to retrieve custom metrics using the `collectd` protocol. You use `collectd` software to send the metrics to the CloudWatch agent. For more information, see [Retrieve Custom Metrics with collectd \(p. 303\)](#).
- `cpu` – Optional. Specifies that CPU metrics are to be collected. This section is valid only for Linux instances. You must include at least one of the `resources` and `totalcpu` fields for any CPU metrics to be collected. This section can include the following fields:
 - `resources` – Optional. Specify this field with a value of `*` to cause per-cpu metrics are to be collected. The only allowed value is `*`.
 - `totalcpu` – Optional. Specifies whether to report cpu metrics aggregated across all cpu cores. The default is `true`.
 - `measurement` – Specifies the array of cpu metrics to be collected. Possible values are `time_active`, `time_guest`, `time_guest_nice`, `time_idle`, `time_iowait`, `time_irq`, `time_nice`, `time_softirq`, `time_steal`, `time_system`, `time_user`, `usage_active`, `usage_guest`, `usage_guest_nice`, `usage_idle`, `usage_iowait`, `usage_irq`, `usage_nice`, `usage_softirq`, `usage_steal`, `usage_system`, and `usage_user`. This field is required if you include `cpu`.

By default, the unit for `cpu_usage_*` metrics is `Percent`, and `cpu_time_*` metrics don't have a unit.

Within the entry for each individual metric, you might optionally specify one or both of the following:

- `rename` – Specifies a different name for this metric.
- `unit` – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- `metrics_collection_interval` – Optional. Specifies how often to collect the cpu metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

This value is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Additional dimensions to use for only the `cpu` metrics. If you specify this field, it's used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics that the agent collects.
- `disk` – Optional. Specifies that disk metrics are to be collected. This section is valid only for Linux instances. This section can include as many as two fields:
 - `resources` – Optional. Specifies an array of disk mount points. This field limits CloudWatch to collect metrics from only the listed mount points. You can specify `*` as the value to collect metrics from all mount points. The default value is to collect metrics from all mount points.
 - `measurement` – Specifies the array of disk metrics to be collected. Possible values are `free`, `total`, `used`, `used_percent`, `inodes_free`, `inodes_used`, and `inodes_total`. This field is required if you include `disk`.

Note

The `disk` metrics have a dimension for `Partition`, which means that the number of custom metrics generated is dependent on the number of partitions associated with your instance. The number of disk partitions you have depends on which AMI you are using and the number of Amazon EBS volumes you attach to the server.

To see the default units for each `disk` metric, see [Metrics Collected by the CloudWatch Agent on Linux Instances \(p. 305\)](#).

Within the entry for each individual metric, you might optionally specify one or both of the following:

- `rename` – Specifies a different name for this metric.
- `unit` – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- `ignore_file_system_types` – Specifies file system types to exclude when collecting disk metrics. Valid values include `sysfs`, `devtmpfs`, and so on.
- `drop_device` – Setting this to `true` causes `Device` to not be included as a dimension for disk metrics.

Preventing `Device` from being used as a dimension can be useful on instances that use the Nitro system because on those instances the device names change for each disk mount when the instance is rebooted. This can cause inconsistent data in your metrics and cause alarms based on these metrics to go to `INSUFFICIENT DATA` state.

The default is `false`.

- `metrics_collection_interval` – Optional. Specifies how often to collect the disk metrics, overriding the global `metrics_collection_interval` specified in the agent section of the configuration file.

This value is specified in seconds.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Additional dimensions to use for only the disk metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.
- `diskio` – Optional. Specifies that disk i/o metrics are to be collected. This section is valid only for Linux instances. This section can include as many as two fields:

- **resources** – Optional. If you specify an array of devices, CloudWatch collects metrics from only those devices. Otherwise, metrics for all devices are collected. You can also specify `*` as the value to collect metrics from all devices.
- **measurement** – Specifies the array of diskio metrics to be collected. Possible values are `reads`, `writes`, `read_bytes`, `write_bytes`, `read_time`, `write_time`, `io_time`, and `iops_in_progress`. This field is required if you include `diskio`.

Within the entry for each individual metric, you might optionally specify one or both of the following:

- **rename** – Specifies a different name for this metric.
- **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- **metrics_collection_interval** – Optional. Specifies how often to collect the diskio metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

This value is specified in seconds.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

- **append_dimensions** – Optional. Additional dimensions to use for only the diskio metrics. If you specify this field, it is used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.
- **swap** – Optional. Specifies that swap memory metrics are to be collected. This section is valid only for Linux instances. This section can include as many as three fields:
- **measurement** – Specifies the array of swap metrics to be collected. Possible values are `free`, `used`, and `used_percent`. This field is required if you include `swap`.

To see the default units for each swap metric, see [Metrics Collected by the CloudWatch Agent on Linux Instances \(p. 305\)](#).

Within the entry for each individual metric, you might optionally specify one or both of the following:

- **rename** – Specifies a different name for this metric.
- **unit** – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- **metrics_collection_interval** – Optional. Specifies how often to collect the swap metrics, overriding the global `metrics_collection_interval` specified in the `agent` section of the configuration file.

This value is specified in seconds.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

- **append_dimensions** – Optional. Additional dimensions to use for only the swap metrics. If you specify this field, it is used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics collected by the agent. It's collected as a high-resolution metric.
- **mem** – Optional. Specifies that memory metrics are to be collected. This section is valid only for Linux instances. This section can include as many as three fields:
- **measurement** – Specifies the array of memory metrics to be collected. Possible values are `active`, `available`, `available_percent`, `buffered`, `cached`, `free`, `inactive`, `total`, `used`, and `used_percent`. This field is required if you include `mem`.

To see the default units for each mem metric, see [Metrics Collected by the CloudWatch Agent on Linux Instances \(p. 305\)](#).

Within the entry for each individual metric, you might optionally specify one or both of the following:

- `rename` – Specifies a different name for this metric.
- `unit` – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- `metrics_collection_interval` – Optional. Specifies how often to collect the mem metrics, overriding the global `metrics_collection_interval` specified in the agent section of the configuration file.

This value is specified in seconds.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Additional dimensions to use for only the mem metrics. If you specify this field, it's used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics that the agent collects.
- `net` – Optional. Specifies that networking metrics are to be collected. This section is valid only for Linux instances. This section can include as many as four fields:
 - `resources` – Optional. If you specify an array of network interfaces, CloudWatch collects metrics from only those interfaces. Otherwise, metrics for all devices are collected. You can also specify `*` as the value to collect metrics from all interfaces.
 - `measurement` – Specifies the array of networking metrics to be collected. Possible values are `bytes_sent`, `bytes_recv`, `drop_in`, `drop_out`, `err_in`, `err_out`, `packets_sent`, and `packets_recv`. This field is required if you include `net`.

To see the default units for each net metric, see [Metrics Collected by the CloudWatch Agent on Linux Instances \(p. 305\)](#).

Within the entry for each individual metric, you might optionally specify one or both of the following:

- `rename` – Specifies a different name for this metric.
- `unit` – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- `metrics_collection_interval` – Optional. Specifies how often to collect the net metrics, overriding the global `metrics_collection_interval` specified in the agent section of the configuration file.

This value is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Additional dimensions to use for only the net metrics. If you specify this field, it's used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.
- `netstat` – Optional. Specifies that TCP connection state and UDP connection metrics are to be collected. This section is valid only for Linux instances. This section can include as many as three fields:
 - `measurement` – Specifies the array of netstat metrics to be collected. Possible values are `tcp_close`, `tcp_close_wait`, `tcp_closing`, `tcp_established`, `tcp_fin_wait1`,

tcp_fin_wait2, tcp_last_ack, tcp_listen, tcp_none, tcp_syn_sent, tcp_syn_recv, tcp_time_wait, and udp_socket. This field is required if you include netstat.

To see the default units for each netstat metric, see [Metrics Collected by the CloudWatch Agent on Linux Instances \(p. 305\)](#).

Within the entry for each individual metric, you might optionally specify one or both of the following:

- `rename` – Specifies a different name for this metric.
- `unit` – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- `metrics_collection_interval` – Optional. Specifies how often to collect the netstat metrics, overriding the global `metrics_collection_interval` specified in the agent section of the configuration file.

This value is specified in seconds.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information about high-resolution metrics, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Additional dimensions to use for only the netstat metrics. If you specify this field, it's used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.
- `processes` – Optional. Specifies that process metrics are to be collected. This section is valid only for Linux instances. This section can include as many as three fields:
 - `measurement` – Specifies the array of processes metrics to be collected. Possible values are `blocked`, `dead`, `idle`, `paging`, `running`, `sleeping`, `stopped`, `total`, `total_threads`, `wait`, and `zombies`. This field is required if you include `processes`.

For all `processes` metrics, the default unit is `Count`.

Within the entry for each individual metric, you might optionally specify one or both of the following:

- `rename` – Specifies a different name for this metric.
- `unit` – Specifies the unit to use for this metric, overriding the default unit for the metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).
- `metrics_collection_interval` – Optional. Specifies how often to collect the processes metrics, overriding the global `metrics_collection_interval` specified in the agent section of the configuration file.

This value is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Additional dimensions to use for only the process metrics. If you specify this field, it's used in addition to dimensions specified in the `append_dimensions` field that is used for all types of metrics collected by the agent.
- `procstat` – Optional. Specifies that you want to retrieve metrics from individual processes. For more information, see [Collect Process Metrics with the procstat Plugin \(p. 295\)](#).
- `statsd` – Optional. Specifies that you want to retrieve custom metrics using the StatsD protocol. The CloudWatch agent acts as a daemon for the protocol. You use any standard StatsD client to send the metrics to the CloudWatch agent. For more information, see [Retrieve Custom Metrics with StatsD \(p. 302\)](#).

The following is an example of a `metrics` section for a Linux server. In this example, three CPU metrics, three netstat metrics, three process metrics, and one disk metric are collected, and the agent is set up to receive additional metrics from a `collectd` client.

```
"metrics": {
  "metrics_collected": {
    "collectd": {},
    "cpu": {
      "resources": [
        "*"
      ],
      "measurement": [
        { "name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent" },
        { "name": "cpu_usage_nice", "unit": "Percent" },
        "cpu_usage_guest"
      ],
      "totalcpu": false,
      "metrics_collection_interval": 10,
      "append_dimensions": {
        "test": "test1",
        "date": "2017-10-01"
      }
    },
    "netstat": {
      "measurement": [
        "tcp_established",
        "tcp_syn_sent",
        "tcp_close"
      ],
      "metrics_collection_interval": 60
    },
    "disk": {
      "measurement": [
        "used_percent"
      ],
      "resources": [
        "*"
      ],
      "drop_device": true
    },
    "processes": {
      "measurement": [
        "running",
        "sleeping",
        "dead"
      ]
    }
  },
  "append_dimensions": {
    "ImageId": "${aws:ImageId}",
    "InstanceId": "${aws:InstanceId}",
    "InstanceType": "${aws:InstanceType}",
    "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
  },
  "aggregation_dimensions" : [ ["AutoScalingGroupName"], ["InstanceId", "InstanceType"] ],
[]
}
```

Windows Server

In the `metrics_collected` section for Windows Server, you can have subsections for each Windows performance object, such as Memory, Processor, and LogicalDisk. For information about what objects and counters are available, see the Microsoft Windows documentation.

Within the subsection for each object, you specify a `measurement` array of the counters to collect. The `measurement` array is required for each object that you specify in the configuration file. You can also specify a `resources` field to name the instances to collect metrics from. You can also specify `*` for `resources` to collect separate metrics for every instance. If you omit `resources`, the data for all instances is aggregated into one set. For objects that don't have instances, omit `resources`.

Within each object section, you can also specify the following optional fields:

- `metrics_collection_interval` – Optional. Specifies how often to collect the metrics for this object, overriding the global `metrics_collection_interval` specified in the agent section of the configuration file.

This value is specified in seconds. For example, specifying 10 sets metrics to be collected every 10 seconds, and setting it to 300 specifies metrics to be collected every 5 minutes.

If you set this value below 60 seconds, each metric is collected as a high-resolution metric. For more information, see [High-Resolution Metrics \(p. 50\)](#).

- `append_dimensions` – Optional. Specifies additional dimensions to use for only the metrics for this object. If you specify this field, it's used in addition to dimensions specified in the global `append_dimensions` field that is used for all types of metrics collected by the agent.

Within each counter section, you can also specify the following optional fields:

- `rename` – Specifies a different name to be used in CloudWatch for this metric.
- `unit` – Specifies the unit to use for this metric. The unit that you specify must be a valid CloudWatch metric unit, as listed in the `Unit` description in [MetricDatum](#).

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

- `statsd` – Enables you to retrieve custom metrics using the `StatsD` protocol. The CloudWatch agent acts as a daemon for the protocol. You use any standard `StatsD` client to send the metrics to the CloudWatch agent. For more information, see [Retrieve Custom Metrics with StatsD \(p. 302\)](#).
- `procstat` – Enables you to retrieve metrics from individual processes. For more information, see [Collect Process Metrics with the procstat Plugin \(p. 295\)](#).

The following is an example `metrics` section for use on Windows Server. In this example, many Windows metrics are collected, and the computer is also set to receive additional metrics from a `StatsD` client.

```
"metrics": {
  "metrics_collected": {
    "statsd": {},
    "Processor": {
      "measurement": [
        { "name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
        "% Interrupt Time",
        "% User Time",
        "% Processor Time"
      ],
      "resources": [
        "*"
      ],
      "append_dimensions": {
        "d1": "win_foo",
        "d2": "win_bar"
      }
    },
    "LogicalDisk": {
```

```
    "measurement": [
      { "name": "% Idle Time", "unit": "Percent" },
      { "name": "% Disk Read Time", "rename": "DISK_READ" },
      { "name": "% Disk Write Time" }
    ],
    "resources": [
      "*"
    ]
  },
  "Memory": {
    "metrics_collection_interval": 5,
    "measurement": [
      "Available Bytes",
      "Cache Faults/sec",
      "Page Faults/sec",
      "Pages/sec"
    ],
    "append_dimensions": {
      "d3": "win_bo"
    }
  },
  "Network Interface": {
    "metrics_collection_interval": 5,
    "measurement": [
      "Bytes Received/sec",
      "Bytes Sent/sec",
      "Packets Received/sec",
      "Packets Sent/sec"
    ],
    "resources": [
      "*"
    ],
    "append_dimensions": {
      "d3": "win_bo"
    }
  },
  "System": {
    "measurement": [
      "Context Switches/sec",
      "System Calls/sec",
      "Processor Queue Length"
    ],
    "append_dimensions": {
      "d1": "win_foo",
      "d2": "win_bar"
    }
  }
},
"append_dimensions": {
  "ImageId": "${aws:ImageId}",
  "InstanceId": "${aws:InstanceId}",
  "InstanceType": "${aws:InstanceType}",
  "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
},
"aggregation_dimensions" : [ ["ImageId"], ["InstanceId", "InstanceType"], ["d1"], [] ]
}
```

CloudWatch Agent Configuration File: Logs Section

The logs section includes the following fields:

- `logs_collected` – Required if the logs section is included. Specifies which log files and Windows event logs are to be collected from the server. It can include two fields, `files` and `windows_events`.

- `files` – Specifies which regular log files the CloudWatch agent is to collect. It contains one field, `collect_list`, which further defines these files.
- `collect_list` – Required if `files` is included. Contains an array of entries, each of which specifies one log file to collect. Each of these entries can include the following fields:
 - `file_path` – Specifies the path of the log file to upload to CloudWatch Logs. Standard Unix glob matching rules are accepted, with the addition of `**` as a *super asterisk*. For example, specifying `/var/log/**/*.log` causes all `.log` files in the `/var/log` directory tree to be collected. For more examples, see [Glob Library](#).

You can also use the standard asterisk as a standard wildcard. For example, `/var/log/system.log*` matches files such as `system.log_1111`, `system.log_2222`, and so on in `/var/log`.

Only the latest file is pushed to CloudWatch Logs based on file modification time. We recommend that you use wildcards to specify a series of files of the same type, such as `access_log.2018-06-01-01` and `access_log.2018-06-01-02`, but not multiple kinds of files, such as `access_log_80` and `access_log_443`. To specify multiple kinds of files, add another log stream entry to the agent configuration file so that each kind of log file goes to a different log stream.

- `auto_removal` – Optional. If this is true, the CloudWatch agent automatically removes old log files after they are uploaded to CloudWatch Logs. The agent only removes complete files from logs that create multiple files, such as logs that create separate files for each date. If a log continuously writes to a single file, it is not removed.

If you already have a log file rotation or removal method in place, we recommend that you omit this field or set it to `false`.

If you omit this field, the default value of `false` is used.

- `log_group_name` – Optional. Specifies what to use as the log group name in CloudWatch Logs. As part of the name, you can use `{instance_id}`, `{hostname}`, `{local_hostname}`, and `{ip_address}` as variables within the name. `{hostname}` retrieves the hostname from the EC2 metadata, and `{local_hostname}` uses the hostname from the network configuration file.

If you use these variables to create many different log groups, keep in mind the limit of 5000 log groups per account per Region.

Allowed characters include a–z, A–Z, 0–9, `_` (underscore), `-` (hyphen), `/` (forward slash), and `.` (period).

We recommend that you specify this field to prevent confusion. If you omit this field, the file path up to the final dot is used as the log group name. For example, if the file path is `/tmp/TestLogFile.log.2017-07-11-14`, the log group name is `/tmp/TestLogFile.log`.

- `log_stream_name` – Optional. Specifies what to use as the log stream name in CloudWatch Logs. As part of the name, you can use `{instance_id}`, `{hostname}`, `{local_hostname}`, and `{ip_address}` as variables within the name. `{hostname}` retrieves the hostname from the EC2 metadata, and `{local_hostname}` uses the hostname from the network configuration file.

If you omit this field, the default value of `{instance_id}` is used. If a log stream doesn't already exist, it's created automatically.

- `timezone` – Optional. Specifies the time zone to use when putting timestamps on log events. The valid values are `UTC` and `Local`. The default value is `Local`.

This parameter is ignored if you don't specify a value for `timestamp_format`.

- `timestamp_format` – Optional. Specifies the timestamp format, using plaintext and special symbols that start with `%`. If you omit this field, the current time is used. If you use this field, you can use the symbols in the following list as part of the format.

If a single log entry contains two time stamps that match the format, the first time stamp is used.

This list of symbols is different than the list used by the older CloudWatch Logs agent. For a summary of these differences, see [Timestamp Differences Between the Unified CloudWatch Agent and the Older CloudWatch Logs Agent \(p. 320\)](#)

`%y`

Year without century as a zero-padded decimal number. For example, 19 to represent 2019.

`%Y`

Year with century as a decimal number. For example, 2019.

`%b`

Month as the locale's abbreviated name

`%B`

Month as the locale's full name

`%m`

Month as a zero-padded decimal number

`%-m`

Month as a decimal number (not zero-padded)

`%d`

Day of the month as a zero-padded decimal number

`%-d`

Day of the month as a decimal number (not zero-padded)

`%A`

Full name of weekday, such as Monday

`%a`

Abbreviation of weekday, such as Mon

`%H`

Hour (in a 24-hour clock) as a zero-padded decimal number

`%I`

Hour (in a 12-hour clock) as a zero-padded decimal number

`%-I`

Hour (in a 12-hour clock) as a decimal number (not zero-padded)

`%p`

AM or PM

`%M`

Minutes as a zero-padded decimal number

`%-M`

Minutes as a decimal number (not zero-padded)

`%S`

Seconds as a zero-padded decimal number

`%-S`

Seconds as a decimal number (not zero padded)

`%f`

Fractional seconds as a decimal number (1-9 digits), zero-padded on the left.

`%Z`

Time zone, for example PST

`%z`

Time zone, expressed as the offset between the local time zone and UTC. For example, -0700. Only this format is supported. For example, -07:00 isn't a valid format.

- `multi_line_start_pattern` – Specifies the pattern for identifying the start of a log message. A log message is made of a line that matches the pattern and any subsequent lines that don't match the pattern.

If you omit this field, multi-line mode is disabled, and any line that begins with a non-whitespace character closes the previous log message and starts a new log message.

If you include this field, you can specify `{timestamp_format}` to use the same regular expression as your timestamp format. Otherwise, you can specify a different regular expression for CloudWatch Logs to use to determine the start lines of multi-line entries.

- `encoding` – Specified the encoding of the log file so that it can be read correctly. If you specify an incorrect coding, there might be data loss because characters that can't be decoded are replaced with other characters.

The default value is `utf-8`. The following are all possible values:

`ascii, big5, euc-jp, euc-kr, gbk, gb18030, ibm866, iso2022-jp, iso8859-2, iso8859-3, iso8859-4, iso8859-5, iso8859-6, iso8859-7, iso8859-8, iso8859-8-i, iso8859-10, iso8859-13, iso8859-14, iso8859-15, iso8859-16, koi8-r, koi8-u, macintosh, shift_jis, utf-8, utf-16, windows-874, windows-1250, windows-1251, windows-1252, windows-1253, windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, x-mac-cyrillic`

- The `windows_events` section specifies the type of Windows events to collect from servers running Windows Server. It includes the following fields:
 - `collect_list` – Required if `windows_events` is included. Specifies the types and levels of Windows events to be collected. Each log to be collected has an entry in this section, which can include the following fields:
 - `event_name` – Specifies the type of Windows events to log. This is equivalent to the Windows event log channel name: for example, System, Security, Application, and so on. This field is required for each type of Windows event to log.
 - `event_levels` – Specifies the levels of event to log. You must specify each level to log. Possible values include INFORMATION, WARNING, ERROR, CRITICAL, and VERBOSE. This field is required for each type of Windows event to log.
 - `log_group_name` – Required. Specifies what to use as the log group name in CloudWatch Logs.

- `log_stream_name` – Optional. Specifies what to use as the log stream name in CloudWatch Logs. As part of the name, you can use `{instance_id}`, `{hostname}`, `{local_hostname}`, and `{ip_address}` as variables within the name. `{hostname}` retrieves the hostname from the EC2 metadata, and `{local_hostname}` uses the hostname from the network configuration file.

If you omit this field, the default value of `{instance_id}` is used. If a log stream doesn't already exist, it's created automatically.

- `event_format` – Optional. Specifies the format to use when storing Windows events in CloudWatch Logs. `xml` uses the XML format as in Windows Event Viewer. `text` uses the legacy CloudWatch Logs agent format.
- `log_stream_name` – Required. Specifies the default log stream name to be used for any logs or Windows events that don't have individual log stream names defined in their entry in `collect_list`.
- `endpoint_override` – Specifies a FIPS endpoint or private link to use as the endpoint where the agent sends logs. Specifying this field and setting a private link enables you to send the logs to an Amazon VPC endpoint. For more information, see [What Is Amazon VPC?](#)

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

For example, the following part of the logs section of the configuration file sets the agent to use a VPC Endpoint when sending logs.

```
{
  "logs": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXX.monitoring.us-
east-1.vpce.amazonaws.com",
    .....
  },
}
```

- `force_flush_interval` – Specifies in seconds the maximum amount of time that logs remain in the memory buffer before being sent to the server. No matter the setting for this field, if the size of the logs in the buffer reaches 1 MB, the logs are immediately sent to the server. The default value is 5.
- `credentials` – Specifies an IAM role to use when sending logs to a different AWS account. If specified, this field contains one parameter, `role_arn`.
 - `role_arn` – Specifies the ARN of an IAM role to use for authentication when sending logs to a different AWS account. For more information, see [Sending Metrics and Logs to a Different Account \(p. 318\)](#). If specified here, this overrides the `role_arn` specified in the agent section of the configuration file, if any.
- `metrics_collected` – Specifies that the agent is to collect metrics embedded in logs. Currently, the `metrics_collected` field can contain only the `emf` field.
 - `emf` – Specifies that the agent is to collect logs that are in embedded metric format. You can generate metric data from these logs. For more information, see [Ingesting High-Cardinality Logs and Generating Metrics with CloudWatch Embedded Metric Format \(p. 387\)](#).

The following is an example of a logs section.

```
"logs":
{
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\
\\amazon-cloudwatch-agent.log",
          "log_group_name": "amazon-cloudwatch-agent.log",
          "log_stream_name": "my_log_stream_name_1",
```

```
        "timestamp_format": "%H: %M: %S%y%b%-d"
      },
      {
        "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\
\\test.log",
        "log_group_name": "test.log",
        "log_stream_name": "my_log_stream_name_2"
      }
    ]
  },
  "windows_events": {
    "collect_list": [
      {
        "event_name": "System",
        "event_levels": [
          "INFORMATION",
          "ERROR"
        ],
        "log_group_name": "System",
        "log_stream_name": "System"
      },
      {
        "event_name": "CustomizedName",
        "event_levels": [
          "INFORMATION",
          "ERROR"
        ],
        "log_group_name": "CustomizedLogGroup",
        "log_stream_name": "CustomizedLogStream"
      }
    ]
  }
},
"log_stream_name": "my_log_stream_name"
}
```

CloudWatch Agent Configuration File: Complete Examples

The following is an example of a complete CloudWatch agent configuration file for a Linux server.

The items listed in the measurement sections for the metrics you want to collect can either specify the complete metric name such or just the part of the metric name that will be appended to the type of resource. For example, specifying either reads or diskio_reads in the measurement section of the diskio section will cause the diskio_reads metric to be collected.

This example includes both ways of specifying metrics in the measurement section.

```
{
  "agent": {
    "metrics_collection_interval": 10,
    "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
  },
  "metrics": {
    "metrics_collected": {
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          { "name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent" },
          { "name": "cpu_usage_nice", "unit": "Percent" },
          "cpu_usage_guest"
        ]
      }
    }
  }
}
```

```
"totalcpu": false,
"metrics_collection_interval": 10,
"append_dimensions": {
  "customized_dimension_key_1": "customized_dimension_value_1",
  "customized_dimension_key_2": "customized_dimension_value_2"
},
},
"disk": {
  "resources": [
    "/",
    "/tmp"
  ],
  "measurement": [
    {"name": "free", "rename": "DISK_FREE", "unit": "Gigabytes"},
    "total",
    "used"
  ],
  "ignore_file_system_types": [
    "sysfs", "devtmpfs"
  ],
  "metrics_collection_interval": 60,
  "append_dimensions": {
    "customized_dimension_key_3": "customized_dimension_value_3",
    "customized_dimension_key_4": "customized_dimension_value_4"
  }
},
"diskio": {
  "resources": [
    "*"
  ],
  "measurement": [
    "reads",
    "writes",
    "read_time",
    "write_time",
    "io_time"
  ],
  "metrics_collection_interval": 60
},
"swap": {
  "measurement": [
    "swap_used",
    "swap_free",
    "swap_used_percent"
  ]
},
"mem": {
  "measurement": [
    "mem_used",
    "mem_cached",
    "mem_total"
  ],
  "metrics_collection_interval": 1
},
"net": {
  "resources": [
    "eth0"
  ],
  "measurement": [
    "bytes_sent",
    "bytes_recv",
    "drop_in",
    "drop_out"
  ]
},
"netstat": {
```

```

    "measurement": [
      "tcp_established",
      "tcp_syn_sent",
      "tcp_close"
    ],
    "metrics_collection_interval": 60
  },
  "processes": {
    "measurement": [
      "running",
      "sleeping",
      "dead"
    ]
  }
},
"append_dimensions": {
  "ImageId": "${aws:ImageId}",
  "InstanceId": "${aws:InstanceId}",
  "InstanceType": "${aws:InstanceType}",
  "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
},
"aggregation_dimensions" : [ ["ImageId"], ["InstanceId", "InstanceType"], ["d1"] ],
[[]],
  "force_flush_interval" : 30
},
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
          "log_group_name": "amazon-cloudwatch-agent.log",
          "log_stream_name": "amazon-cloudwatch-agent.log",
          "timezone": "UTC"
        },
        {
          "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
          "log_group_name": "test.log",
          "log_stream_name": "test.log",
          "timezone": "Local"
        }
      ]
    }
  },
  "log_stream_name": "my_log_stream_name",
  "force_flush_interval" : 15
}
}

```

The following is an example of a complete CloudWatch agent configuration file for a server running Windows Server.

```

{
  "agent": {
    "metrics_collection_interval": 60,
    "logfile": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log"
  },
  "metrics": {
    "metrics_collected": {
      "Processor": {
        "measurement": [
          { "name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent" },

```

```
        "% Interrupt Time",
        "% User Time",
        "% Processor Time"
    ],
    "resources": [
        "*"
    ],
    "append_dimensions": {
        "customized_dimension_key_1": "customized_dimension_value_1",
        "customized_dimension_key_2": "customized_dimension_value_2"
    }
},
"LogicalDisk": {
    "measurement": [
        {"name": "% Idle Time", "unit": "Percent"},
        {"name": "% Disk Read Time", "rename": "DISK_READ"},
        "% Disk Write Time"
    ],
    "resources": [
        "*"
    ]
},
"customizedObjectName": {
    "metrics_collection_interval": 60,
    "customizedCounterName": [
        "metric1",
        "metric2"
    ],
    "resources": [
        "customizedInstances"
    ]
},
"Memory": {
    "metrics_collection_interval": 5,
    "measurement": [
        "Available Bytes",
        "Cache Faults/sec",
        "Page Faults/sec",
        "Pages/sec"
    ]
},
"Network Interface": {
    "metrics_collection_interval": 5,
    "measurement": [
        "Bytes Received/sec",
        "Bytes Sent/sec",
        "Packets Received/sec",
        "Packets Sent/sec"
    ],
    "resources": [
        "*"
    ],
    "append_dimensions": {
        "customized_dimension_key_3": "customized_dimension_value_3"
    }
},
"System": {
    "measurement": [
        "Context Switches/sec",
        "System Calls/sec",
        "Processor Queue Length"
    ]
}
},
"append_dimensions": {
    "ImageId": "${aws:ImageId}",
```

```
    "InstanceId": "${aws:InstanceId}",
    "InstanceType": "${aws:InstanceType}",
    "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
  },
  "aggregation_dimensions" : [ ["ImageId"], ["InstanceId", "InstanceType"], ["d1"], [] ]
},
"logs": {
  "logs_collected": {
    "files": {
      "collect_list": [
        {
          "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log",
          "log_group_name": "amazon-cloudwatch-agent.log",
          "timezone": "UTC"
        },
        {
          "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\
\\test.log",
          "log_group_name": "test.log",
          "timezone": "Local"
        }
      ]
    }
  },
  "windows_events": {
    "collect_list": [
      {
        "event_name": "System",
        "event_levels": [
          "INFORMATION",
          "ERROR"
        ],
        "log_group_name": "System",
        "log_stream_name": "System",
        "event_format": "xml"
      },
      {
        "event_name": "CustomizedName",
        "event_levels": [
          "WARNING",
          "ERROR"
        ],
        "log_group_name": "CustomizedLogGroup",
        "log_stream_name": "CustomizedLogStream",
        "event_format": "xml"
      }
    ]
  }
},
"log_stream_name": "example_log_stream_name"
}
```

Save the CloudWatch Agent Configuration File Manually

If you create or edit the CloudWatch agent configuration file manually, you can give it any name. For simplicity in troubleshooting, we recommend that you name it `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json` on a Linux server and `$Env:ProgramData\\Amazon\\AmazonCloudWatchAgent\\amazon-cloudwatch-agent.json` on servers running Windows Server. After you have created the file, you can copy it to other servers where you want to run the agent.

Uploading the CloudWatch Agent Configuration File to Systems Manager Parameter Store

If you plan to use the SSM Agent to install the CloudWatch agent on servers, after you manually edit the CloudWatch agent configuration file, you can upload it to Systems Manager Parameter Store. To do so, use the Systems Manager `put-parameter` command.

To be able to store the file in Parameter Store, you must use an IAM role with sufficient permissions. For more information, see [Create IAM Roles and Users for Use with the CloudWatch Agent \(p. 249\)](#).

Use the following command, where *parameter name* is the name to be used for this file in Parameter Store and *configuration_file_pathname* is the path and file name of the configuration file that you edited.

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

Collect Process Metrics with the procstat Plugin

The *procstat* plugin enables you to collect metrics from individual processes. It is supported on Linux servers and on servers running Windows Server 2008 R2 or later.

Topics

- [Configuring the CloudWatch Agent for procstat \(p. 295\)](#)
- [Metrics Collected by Procstat \(p. 297\)](#)

Configuring the CloudWatch Agent for procstat

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

- *pid_file*: Selects processes by the names of the process identification number (PID) files they create.
- *exe*: Selects the processes that have process names that match the string that you specify, using regular expression matching rules. The match is a "contains" match, meaning that if you specify *agent* as the term to match, processes with names like *cloudwatchagent* match the term. For more information, see [Syntax](#).
- *pattern*: Selects processes by the command lines used to start the processes. All processes are selected that have command lines matching the specified string using regular expression matching rules. The entire command line is checked, including parameters and options used with the command.

The match is a "contains" match, meaning that if you specify *-c* as the term to match, processes with parameters like *-config* match the term.

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

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

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

In the examples in the following sections, the `procstat` section is the only section included in the `metrics_collected` section of the agent configuration file. Actual configuration files can also include other sections in `metrics_collected`. For more information, see [Manually Create or Edit the CloudWatch Agent Configuration File \(p. 275\)](#).

Configuring with `Pid_file`

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

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

Configuring with `Exe`

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

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



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

Configuring with Pattern

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

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

Metrics Collected by Procstat

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

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

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

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

Metric Name	Available On	Description
memory_stack	Linux	The amount of stack memory that the process is using. Unit: Bytes
memory_swap	Linux	The amount of swap memory that the process is using. Unit: Bytes
memory_vms	Linux, Windows Server	The amount of virtual memory that the process is using. Unit: Bytes
pid	Linux, Windows Server	Process identifier (ID). Unit: Count
pid_count	Linux, Windows Server	The number of process IDs associated with the process. On Linux servers the full name of this metric is <code>procstat_lookup_pid_count</code> and on Windows Server it is <code>procstat_lookup_pid_count</code> . Unit: Count
read_bytes	Linux, Windows Server	The number of bytes that the process has read from disks. Unit: Bytes
write_bytes	Linux, Windows Server	The number of bytes that the process has written to disks. Unit: Bytes

Metric Name	Available On	Description
<code>read_count</code>	Linux, Windows Server	The number of disk read operations that the process has executed. Unit: Count
<code>write_count</code>	Linux, Windows Server	The number of disk write operations that the process has executed. Unit: Count
<code>involuntary_context_switches</code>	Linux	The number of times that the process was involuntarily context-switched. Unit: Count
<code>voluntary_context_switches</code>	Linux	The number of times that the process was context-switched voluntarily. Unit: Count
<code>realtime_priority</code>	Linux	The current usage of real-time priority for the process. Unit: Count
<code>nice_priority</code>	Linux	The current usage of nice priority for the process. Unit: Count
<code>signals_pending</code>	Linux	The number of signals pending to be handled by the process. Unit: Count

Metric Name	Available On	Description
<code>rlimit_cpu_time_hard</code>	Linux	The hard CPU time resource limit for the process. Unit: Count
<code>rlimit_cpu_time_soft</code>	Linux	The soft CPU time resource limit for the process. Unit: Count
<code>rlimit_file_locks_hard</code>	Linux	The hard file locks resource limit for the process. Unit: Count
<code>rlimit_file_locks_soft</code>	Linux	The soft file locks resource limit for the process. Unit: Count
<code>rlimit_memory_data_hard</code>	Linux	The hard resource limit on the process for memory used for data. Unit: Bytes
<code>rlimit_memory_data_soft</code>	Linux	The soft resource limit on the process for memory used for data. Unit: Bytes
<code>rlimit_memory_locked_hard</code>	Linux	The hard resource limit on the process for locked memory. Unit: Bytes
<code>rlimit_memory_locked_soft</code>	Linux	The soft resource limit on the process for locked memory. Unit: Bytes

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

Retrieve Custom Metrics with StatsD

You can retrieve custom metrics from your applications or services using the CloudWatch agent with the StatsD protocol. StatsD is supported on both Linux servers and servers running Windows Server. CloudWatch supports the following StatsD format:

```
MetricName:value|type|@sample_rate|#tag1:  
value,tag1...
```

- *MetricName* – A string with no colons, bars, # characters, or @ characters.
- *value* – This can be either integer or float.
- *type* – Specify *c* for counter, *g* for gauge, *ms* for timer, *h* for histogram, or *s* for set.
- *sample_rate* – (Optional) A float between 0 and 1, inclusive. Use only for counter, histogram, and timer metrics. The default value is 1 (sampling 100% of the time).
- *tags* – (Optional) A comma-separated list of tags. StatsD tags are similar to dimensions in CloudWatch. Use colons for key/value tags, such as `env:prod`.

You can use any StatsD client that follows this format to send the metrics to the CloudWatch agent. For more information about some of the available StatsD clients, see the [StatsD client page on GitHub](#).

To collect these custom metrics, add a "statsd": {} line to the metrics_collected section of the agent configuration file. You can add this line manually. If you use the wizard to create the configuration file, it's done for you. For more information, see [Create the CloudWatch Agent Configuration File \(p. 270\)](#).

The StatsD default configuration works for most users. There are three optional fields that you can add to the **statsd** section of the agent configuration file as needed:

- **service_address** – The service address to which the CloudWatch agent should listen. The format is **ip:port**. If you omit the IP address, the agent listens on all available interfaces. Only the UDP format is supported, so you don't need to specify a UDP prefix.

The default value is :8125.

- **metrics_collection_interval** – How often in seconds that the StatsD plugin runs and collects metrics. The default value is 10 seconds. The range is 1–172,000.
- **metrics_aggregation_interval** – How often in seconds CloudWatch aggregates metrics into single data points. The default value is 60 seconds.

For example, if **metrics_collection_interval** is 10 and **metrics_aggregation_interval** is 60, CloudWatch collects data every 10 seconds. After each minute, the six data readings from that minute are aggregated into a single data point, which is sent to CloudWatch.

The range is 0–172,000. Setting **metrics_aggregation_interval** to 0 disables the aggregation of StatsD metrics.

The following is an example of the **statsd** section of the agent configuration file, using the default port and custom collection and aggregation intervals.

```
{
  "metrics":{
    "metrics_collected":{
      "statsd":{
        "service_address":":8125",
        "metrics_collection_interval":60,
        "metrics_aggregation_interval":300
      }
    }
  }
}
```

Retrieve Custom Metrics with collectd

You can retrieve custom metrics from your applications or services using the CloudWatch agent with the collectd protocol, which is supported only on Linux servers. You use the collectd software to send the metrics to the CloudWatch agent. For the collectd metrics, the CloudWatch agent acts as the server while the collectd plug-in acts as the client.

The collectd software is not installed automatically on every server. On a server running Amazon Linux 2, follow these steps to install collectd

```
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum install -y collectd
```

For information about installing collectd on other systems, see the [Download page for collectd](#).

To collect these custom metrics, add a **"collectd": {}** line to the **metrics_collected** section of the agent configuration file. You can add this line manually. If you use the wizard to create the configuration file, it is done for you. For more information, see [Create the CloudWatch Agent Configuration File \(p. 270\)](#).

Optional parameters are also available. If you are using collectd and you do not use `/etc/collectd/auth_file` as your **collectd_auth_file**, you must set some of these options.

- **service_address:** The service address to which the CloudWatch agent should listen. The format is `"udp://ip:port"`. The default is `udp://127.0.0.1:25826`.
- **name_prefix:** A prefix to attach to the beginning of the name of each collectd metric. The default is `collectd_`. The maximum length is 255 characters.
- **collectd_security_level:** Sets the security level for network communication. The default is **encrypt**.

encrypt specifies that only encrypted data is accepted. **sign** specifies that only signed and encrypted data is accepted. **none** specifies that all data is accepted. If you specify a value for **collectd_auth_file**, encrypted data is decrypted if possible.

For more information, see [Client setup](#) and [Possible interactions](#) in the collectd Wiki.

- **collectd_auth_file** Sets a file in which user names are mapped to passwords. These passwords are used to verify signatures and to decrypt encrypted network packets. If given, signed data is verified and encrypted packets are decrypted. Otherwise, signed data is accepted without checking the signature and encrypted data cannot be decrypted.

The default is `/etc/collectd/auth_file`.

If **collectd_security_level** is set to **none**, this is optional. If you set **collectd_security_level** to **encrypt** or **sign**, you must specify **collectd_auth_file**.

For the format of the auth file, each line is a user name followed by a colon and any number of spaces followed by the password. For example:

```
user1: user1_password
```

```
user2: user2_password
```

- **collectd_typesdb:** A list of one or more files that contain the dataset descriptions. The list must be surrounded by brackets, even if there is just one entry in the list. Each entry in the list must be surrounded by double quotes. If there are multiple entries, separate them with commas. The default is `["/usr/share/collectd/types.db"]`. For more information, see <https://collectd.org/documentation/manpages/types.db.5.shtml>.
- **metrics_aggregation_interval:** How often in seconds CloudWatch aggregates metrics into single data points. The default is 60 seconds. The range is 0 to 172,000. Setting it to 0 disables the aggregation of collectd metrics.

The following is an example of the collectd section of the agent configuration file.

```
{
  "metrics":{
    "metrics_collected":{
      "collectd":{
        "name_prefix":"My_collectd_metrics_",
        "metrics_aggregation_interval":120
      }
    }
  }
}
```


Metrics Collected by the CloudWatch Agent

You can collect metrics from servers by installing the CloudWatch agent on the server. You can install the agent on both Amazon EC2 instances and on-premises servers, and on servers running either Linux or Windows Server. If you install the agent on an Amazon EC2 instance, the metrics it collects are in addition to the metrics enabled by default on Amazon EC2 instances.

For information about installing the CloudWatch agent on an instance, see [Collecting Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent](#) (p. 236).

Metrics Collected by the CloudWatch Agent on Windows Server Instances

On a server running Windows Server, installing the CloudWatch agent enables you to collect the metrics associated with the counters in Windows Performance Monitor. The CloudWatch metric names for these counters are created by putting a space between the object name and the counter name. For example, the % Interrupt Time counter of the Processor object is given the metric name `Processor % Interrupt Time` in CloudWatch. For more information about Windows Performance Monitor counters, see the Microsoft Windows Server documentation.

The default namespace for metrics collected by the CloudWatch agent is `CWAgent`, although you can specify a different namespace when you configure the agent.

Metrics Collected by the CloudWatch Agent on Linux Instances

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

Metric	Description
<code>cpu_time_active</code>	The amount of time that the CPU is active in any capacity. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_guest</code>	The amount of time that the CPU is running a virtual CPU for a guest operating system. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_guest_nice</code>	The amount of time that the CPU is running a virtual CPU for a guest operating system, which is low-priority and can be interrupted by other processes. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_idle</code>	The amount of time that the CPU is idle. This metric is measured in hundredths of a second. Unit: None

Metric	Description
<code>cpu_time_iowait</code>	The amount of time that the CPU is waiting for I/O operations to complete. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_irq</code>	The amount of time that the CPU is servicing interrupts. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_nice</code>	The amount of time that the CPU is in user mode with low-priority processes, which can easily be interrupted by higher-priority processes. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_softirq</code>	The amount of time that the CPU is servicing software interrupts. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_steal</code>	The amount of time that the CPU is in <i>stolen time</i> , which is time spent in other operating systems in a virtualized environment. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_system</code>	The amount of time that the CPU is in system mode. This metric is measured in hundredths of a second. Unit: None
<code>cpu_time_user</code>	The amount of time that the CPU is in user mode. This metric is measured in hundredths of a second. Unit: None
<code>cpu_usage_active</code>	The percentage of time that the CPU is active in any capacity. Unit: Percent
<code>cpu_usage_guest</code>	The percentage of time that the CPU is running a virtual CPU for a guest operating system. Unit: Percent
<code>cpu_usage_guest_nice</code>	The percentage of time that the CPU is running a virtual CPU for a guest operating system, which is low-priority and can be interrupted by other processes. Unit: Percent

Metric	Description
<code>cpu_usage_idle</code>	The percentage of time that the CPU is idle. Unit: Percent
<code>cpu_usage_iowait</code>	The percentage of time that the CPU is waiting for I/O operations to complete. Unit: Percent
<code>cpu_usage_irq</code>	The percentage of time that the CPU is servicing interrupts. Unit: Percent
<code>cpu_usage_nice</code>	The percentage of time that the CPU is in user mode with low-priority processes, which higher-priority processes can easily interrupt. Unit: Percent
<code>cpu_usage_softirq</code>	The percentage of time that the CPU is servicing software interrupts. Unit: Percent
<code>cpu_usage_steal</code>	The percentage of time that the CPU is in <i>stolen time</i> , or time spent in other operating systems in a virtualized environment. Unit: Percent
<code>cpu_usage_system</code>	The percentage of time that the CPU is in system mode. Unit: Percent
<code>cpu_usage_user</code>	The percentage of time that the CPU is in user mode. Unit: Percent
<code>disk_free</code>	Free space on the disks. Unit: Bytes
<code>disk_inodes_free</code>	The number of available index nodes on the disk. Unit: Count
<code>disk_inodes_total</code>	The total number of index nodes reserved on the disk. Unit: Count
<code>disk_inodes_used</code>	The number of used index nodes on the disk. Unit: Count
<code>disk_total</code>	Total space on the disks, including used and free. Unit: Bytes

Metric	Description
disk_used	Used space on the disks. Unit: Bytes
disk_used_percent	The percentage of total disk space that is used. Unit: Percent
diskio_iops_in_progress	The number of I/O requests that have been issued to the device driver but have not yet completed. Unit: Count
diskio_io_time	The amount of time that the disk has had I/O requests queued. Unit: Milliseconds The only statistic that should be used for this metric is Sum. Do not use Average.
diskio_reads	The number of disk read operations. Unit: Count The only statistic that should be used for this metric is Sum. Do not use Average.
diskio_read_bytes	The number of bytes read from the disks. Unit: Bytes The only statistic that should be used for this metric is Sum. Do not use Average.
diskio_read_time	The amount of time that read requests have waited on the disks. Multiple read requests waiting at the same time increase the number. For example, if 5 requests all wait for an average of 100 milliseconds, 500 is reported. Unit: Milliseconds The only statistic that should be used for this metric is Sum. Do not use Average.
diskio_writes	The number disk write operations. Unit: Count The only statistic that should be used for this metric is Sum. Do not use Average.
diskio_write_bytes	The number of bytes written to the disks. Unit: Bytes The only statistic that should be used for this metric is Sum. Do not use Average.

Metric	Description
diskio_write_time	<p>The amount of time that write requests have waited on the disks. Multiple write requests waiting at the same time increase the number. For example, if 8 requests all wait for an average of 1000 milliseconds, 8000 is reported.</p> <p>Unit: Milliseconds</p> <p>The only statistic that should be used for this metric is Sum. Do not use Average.</p>
mem_active	<p>The amount of memory that has been used in some way during the last sample period.</p> <p>Unit: Bytes</p>
mem_available	<p>The amount of memory that is available and can be given instantly to processes.</p> <p>Unit: Bytes</p>
mem_available_percent	<p>The percentage of memory that is available and can be given instantly to processes.</p> <p>Unit: Percent</p>
mem_buffered	<p>The amount of memory that is being used for buffers.</p> <p>Unit: Bytes</p>
mem_cached	<p>The amount of memory that is being used for file caches.</p> <p>Unit: Bytes</p>
mem_free	<p>The amount of memory that isn't being used.</p> <p>Unit: Bytes</p>
mem_inactive	<p>The amount of memory that hasn't been used in some way during the last sample period</p> <p>Unit: Bytes</p>
mem_total	<p>The total amount of memory.</p> <p>Unit: Bytes</p>
mem_used	<p>The amount of memory currently in use.</p> <p>Unit: Bytes</p>
mem_used_percent	<p>The percentage of memory currently in use.</p> <p>Unit: Percent</p>

Metric	Description
<code>net_bytes_recv</code>	<p>The number of bytes received by the network interface.</p> <p>Unit: Bytes</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>net_bytes_sent</code>	<p>The number of bytes sent by the network interface.</p> <p>Unit: Bytes</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>net_drop_in</code>	<p>The number of packets received by this network interface that were dropped.</p> <p>Unit: Count</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>net_drop_out</code>	<p>The number of packets transmitted by this network interface that were dropped.</p> <p>Unit: Count</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>net_err_in</code>	<p>The number of receive errors detected by this network interface.</p> <p>Unit: Count</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>net_err_out</code>	<p>The number of transmit errors detected by this network interface.</p> <p>Unit: Count</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>net_packets_sent</code>	<p>The number of packets sent by this network interface.</p> <p>Unit: Count</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>

Metric	Description
<code>net_packets_recv</code>	<p>The number of packets received by this network interface.</p> <p>Unit: Count</p> <p>The only statistic that should be used for this metric is <code>Sum</code>. Do not use <code>Average</code>.</p>
<code>netstat_tcp_close</code>	<p>The number of TCP connections with no state.</p> <p>Unit: Count</p>
<code>netstat_tcp_close_wait</code>	<p>The number of TCP connections waiting for a termination request from the client.</p> <p>Unit: Count</p>
<code>netstat_tcp_closing</code>	<p>The number of TCP connections that are waiting for a termination request with acknowledgement from the client.</p> <p>Unit: Count</p>
<code>netstat_tcp_established</code>	<p>The number of TCP connections established.</p> <p>Unit: Count</p>
<code>netstat_tcp_fin_wait1</code>	<p>The number of TCP connections in the <code>FIN_WAIT1</code> state during the process of closing a connection.</p> <p>Unit: Count</p>
<code>netstat_tcp_fin_wait2</code>	<p>The number of TCP connections in the <code>FIN_WAIT2</code> state during the process of closing a connection.</p> <p>Unit: Count</p>
<code>netstat_tcp_last_ack</code>	<p>The number of TCP connections waiting for the client to send acknowledgement of the connection termination message. This is the last state right before the connection is closed down.</p> <p>Unit: Count</p>
<code>netstat_tcp_listen</code>	<p>The number of TCP ports currently listening for a connection request.</p> <p>Unit: Count</p>
<code>netstat_tcp_none</code>	<p>The number of TCP connections with inactive clients.</p> <p>Unit: Count</p>
<code>netstat_tcp_syn_sent</code>	<p>The number of TCP connections waiting for a matching connection request after having sent a connection request.</p> <p>Unit: Count</p>

Metric	Description
<code>netstat_tcp_syn_recv</code>	The number of TCP connections waiting for connection request acknowledgement after having sent and received a connection request. Unit: Count
<code>netstat_tcp_time_wait</code>	The number of TCP connections currently waiting to ensure that the client received the acknowledgement of its connection termination request. Unit: Count
<code>netstat_udp_socket</code>	The number of current UDP connections. Unit: Count
<code>processes_blocked</code>	The number of processes that are blocked. Unit: Count
<code>processes_dead</code>	The number of processes that are dead, indicated by the <code>X</code> state code on Linux. Unit: Count
<code>processes_idle</code>	The number of processes that are idle (sleeping for more than 20 seconds). Available only on FreeBSD instances. Unit: Count
<code>processes_paging</code>	The number of processes that are paging, indicated by the <code>w</code> state code on Linux. Unit: Count
<code>processes_running</code>	The number of processes that are running, indicated by the <code>R</code> state code. Unit: Count
<code>processes_sleeping</code>	The number of processes that are sleeping, indicated by the <code>S</code> state code. Unit: Count
<code>processes_stopped</code>	The number of processes that are stopped, indicated by the <code>T</code> state code. Unit: Count
<code>processes_total</code>	The total number of processes on the instance. Unit: Count
<code>processes_total_threads</code>	The total number of threads making up the processes. This metric is available only on Linux instances. Unit: Count

Metric	Description
<code>processes_wait</code>	The number of processes that are paging, indicated by the w state code on FreeBSD instances. This metric is available only on FreeBSD instances. Unit: Count
<code>processes_zombies</code>	The number of zombie processes, indicated by the z state code. Unit: Count
<code>swap_free</code>	The amount of swap space that isn't being used. Unit: Bytes
<code>swap_used</code>	The amount of swap space currently in use. Unit: Bytes
<code>swap_used_percent</code>	The percentage of swap space currently in use. Unit: Percent

Common Scenarios with the CloudWatch Agent

The following sections outline how to complete some common configuration and customization tasks when using the CloudWatch agent.

Topics

- [Running the CloudWatch Agent as a Different User \(p. 313\)](#)
- [Adding Custom Dimensions to Metrics Collected by the CloudWatch Agent \(p. 315\)](#)
- [Multiple CloudWatch Agent Configuration Files \(p. 315\)](#)
- [Aggregating or Rolling Up Metrics Collected by the CloudWatch Agent \(p. 317\)](#)
- [Collecting High-Resolution Metrics With the CloudWatch agent \(p. 317\)](#)
- [Sending Metrics and Logs to a Different Account \(p. 318\)](#)
- [Timestamp Differences Between the Unified CloudWatch Agent and the Older CloudWatch Logs Agent \(p. 320\)](#)

Running the CloudWatch Agent as a Different User

On Linux servers, the CloudWatch runs as the root user by default. To have the agent run as a different user, use the `run_as_user` parameter in the `agent` section in the CloudWatch agent configuration file. This option is available only on Linux servers.

If you're already running the agent with the root user and want to change to using a different user, use one of the following procedures.

To run the CloudWatch agent as a different user on an EC2 instance running Linux

1. Download and install a new CloudWatch agent package. For more information, see [Download the CloudWatch Agent Package Using an S3 Download Link \(p. 238\)](#).
2. Create a new Linux user or use the default user named `cwagent` that the RPM or DEB file created.

3. Provide credentials for this user in one of these ways:

- If the file `/home/root/.aws/credentials` exists, you must create a credentials file for the user you are going to use to run the CloudWatch agent. This credentials file will be `/home/username/.aws/credentials`. Then set the value of the `shared_credential_file` parameter in `common-config.toml` to the pathname of the credential file. For more information, see [\(Optional\) Modify the Common Configuration for Proxy or Region Information](#) (p. 247).
- If the file `/home/root/.aws/credentials` does not exist, you can do one of the following:
 - Create a credentials file for the user you are going to use to run the CloudWatch agent. This credentials file will be `/home/username/.aws/credentials`. Then set the value of the `shared_credential_file` parameter in `common-config.toml` to the pathname of the credential file. For more information, see [\(Optional\) Modify the Common Configuration for Proxy or Region Information](#) (p. 247).
 - Instead of creating a credentials file, attach an IAM role to the instance. The agent uses this role as the credential provider.

4. In the CloudWatch agent configuration file, add the following line in the agent section:

```
"run_as_user": "username"
```

Make other modifications to the configuration file as needed. For more information, see [Create the CloudWatch Agent Configuration File](#) (p. 270)

5. Give the user necessary permissions. The user must have Read (r) permissions for the log files to be collected, and must have Execute (x) permission on every directory in the log files' path.
6. Start the agent with the configuration file that you just modified.

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

To run the CloudWatch agent as a different user on an on-premises server running Linux

1. Download and install a new CloudWatch agent package. For more information, see [Download the CloudWatch Agent Package Using an S3 Download Link](#) (p. 238).
2. Create a new Linux user or use the default user named `cwagent` that the RPM or DEB file created.
3. Store the credentials of this user to a path that the user can access, such as `/home/username/.aws/credentials`.
4. Set the value of the `shared_credential_file` parameter in `common-config.toml` to the pathname of the credential file. For more information, see [\(Optional\) Modify the Common Configuration for Proxy or Region Information](#) (p. 247).
5. In the CloudWatch agent configuration file, add the following line in the agent section:

```
"run_as_user": "username"
```

Make other modifications to the configuration file as needed. For more information, see [Create the CloudWatch Agent Configuration File](#) (p. 270)

6. Give the user necessary permissions. The user must have Read (r) permissions for the log files to be collected, and must have Execute (x) permission on every directory in the log files' path.
7. Start the agent with the configuration file that you just modified.

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

Adding Custom Dimensions to Metrics Collected by the CloudWatch Agent

To add custom dimensions such as tags to metrics collected by the agent, add the `append_dimensions` field to the section of the agent configuration file that lists those metrics.

For example, the following example section of the configuration file adds a custom dimension named `stackName` with a value of `Prod` to the `cpu` and `disk` metrics collected by the agent.

```
"cpu":{
  "resources":[
    "*"
  ],
  "measurement":[
    "cpu_usage_guest",
    "cpu_usage_nice",
    "cpu_usage_idle"
  ],
  "totalcpu":false,
  "append_dimensions":{
    "stackName":"Prod"
  }
},
"disk":{
  "resources":[
    "/",
    "/tmp"
  ],
  "measurement":[
    "total",
    "used"
  ],
  "append_dimensions":{
    "stackName":"Prod"
  }
}
```

Remember that any time you change the agent configuration file, you must restart the agent to have the changes take effect.

Multiple CloudWatch Agent Configuration Files

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

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

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

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

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

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

infrastructure.json

```
{  
  "metrics": {  
    "metrics_collected": {  
      "cpu": {  
        "resources": [  
          "*"   
        ],  
        "measurement": [  
          "usage_active"  
        ],  
        "totalcpu": true  
      },  
      "mem": {  
        "measurement": [  
          "used_percent"  
        ]  
      }  
    }  
  },  
  "logs": {  
    "logs_collected": {  
      "files": {  
        "collect_list": [  
          {  
            "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-  
agent.log",  
            "log_group_name": "amazon-cloudwatch-agent.log"  
          },  
          {  
            "file_path": "/var/log/messages",  
            "log_group_name": "/var/log/messages"  
          }  
        ]  
      }  
    }  
  }  
}
```

app.json

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

```
}  
}  
}
```

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

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

Aggregating or Rolling Up Metrics Collected by the CloudWatch Agent

To aggregate or roll up metrics collected by the agent, add an `aggregation_dimensions` field to the section for that metric in the agent configuration file.

For example, the following configuration file snippet rolls up metrics on the `AutoScalingGroupName` dimension. The metrics from all instances in each Auto Scaling group are aggregated and can be viewed as a whole.

```
"metrics": {  
  "cpu":{...}  
  "disk":{...}  
  "aggregation_dimensions" : [ "AutoScalingGroupName" ]  
}
```

To roll up along the combination of each `InstanceId` and `InstanceType` dimensions in addition to rolling up on the Auto Scaling group name, add the following.

```
"metrics": {  
  "cpu":{...}  
  "disk":{...}  
  "aggregation_dimensions" : [ "AutoScalingGroupName", "InstanceId", "InstanceType" ]  
}
```

To roll up metrics into one collection instead, use `[]`.

```
"metrics": {  
  "cpu":{...}  
  "disk":{...}  
  "aggregation_dimensions" : [ [] ]  
}
```

Remember that any time you change the agent configuration file, you must restart the agent to have the changes take effect.

Collecting High-Resolution Metrics With the CloudWatch agent

The `metrics_collection_interval` field specifies the time interval for the metrics collected, in seconds. By specifying a value of less than 60 for this field, the metrics are collected as high-resolution metrics.

For example, if your metrics should all be high-resolution and collected every 10 seconds, specify 10 as the value for `metrics_collection_interval` under the `agent` section as a global metrics collection interval.

```
"agent": {
  "metrics_collection_interval": 10
}
```

Alternatively, the following example sets the `cpu` metrics to be collected every second, and all other metrics are collected every minute.

```
"agent":{
  "metrics_collection_interval": 60
},
"metrics":{
  "metrics_collected":{
    "cpu":{
      "resources":[
        "*"
      ],
      "measurement":[
        "cpu_usage_guest"
      ],
      "totalcpu":false,
      "metrics_collection_interval": 1
    },
    "disk":{
      "resources":[
        "/",
        "/tmp"
      ],
      "measurement":[
        "total",
        "used"
      ]
    }
  }
}
```

Remember that any time you change the agent configuration file, you must restart the agent to have the changes take effect.

Sending Metrics and Logs to a Different Account

To have the CloudWatch agent send the metrics, logs, or both to a different account, specify a `role_arn` parameter in the agent configuration file on the sending server. The `role_arn` value specifies an IAM role in the target account that the agent uses when sending data to the target account. This role enables the sending account to assume a corresponding role in the target account when delivering the metrics or logs to the target account.

You can also specify two separate `role_arn` strings in the agent configuration file: one to use when sending metrics and another for sending logs.

The following example of part of the agent section of the configuration file sets the agent to use `CrossAccountAgentRole` when sending metrics and logs to a different account.

```
{
  "agent": {
    "credentials": {
      "role_arn": "arn:aws:iam::123456789012:role/CrossAccountAgentRole"
    }
  }
}
```

```
    },  
    },  
    .....  
}
```

Alternatively, the following example sets different roles for the sending account to use for sending metrics and logs:

```
"metrics": {  
  "credentials": {  
    "role_arn": "RoleToSendMetrics"  
  },  
  "metrics_collected": {....
```

```
"logs": {  
  "credentials": {  
    "role_arn": "RoleToSendLogs"  
  },  
  ....
```

Policies Needed

When you specify a `role_arn` in the agent configuration file, you must also make sure the IAM roles of the sending and target accounts have certain policies. The roles in both the sending and target accounts should have `CloudWatchAgentServerPolicy`. For more information about assigning this policy to a role, see [Create IAM Roles to Use with the CloudWatch Agent on Amazon EC2 Instances \(p. 249\)](#).

The role in the sending account also must include the following policy. You add this policy on the **Permissions** tab in the IAM console when you edit the role.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "sts:AssumeRole"  
      ],  
      "Resource": [  
        "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"  
      ]  
    }  
  ]  
}
```

The role in the target account must include the following policy so that it recognizes the IAM role used by the sending account. You add this policy on the **Trust relationships** tab in the IAM console when you edit the role. The role in the target account where you add this policy is the role you created in [Create IAM Roles and Users for Use With CloudWatch Agent \(p. 242\)](#). This role is the role specified in *agent-role-in-target-account* in the policy used by the sending account.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": [  
          "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"  
        ]  
      }  
    }  
  ]  
}
```

```
    "arn:aws:iam::sending-account-ID:role/role-in-sender-account"
  },
  "Action": "sts:AssumeRole"
}
]
```

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

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

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

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

Troubleshooting the CloudWatch Agent

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

Topics

- [CloudWatch Agent Command Line Parameters](#) (p. 320)
- [Installing the CloudWatch Agent Using Run Command Fails](#) (p. 321)
- [The CloudWatch Agent Won't Start](#) (p. 321)
- [Verify That the CloudWatch Agent Is Running](#) (p. 321)
- [Where Are the Metrics?](#) (p. 322)
- [The CloudWatch Agent Won't Start, and the Error Mentions an Amazon EC2 Region](#) (p. 322)
- [The CloudWatch Agent Won't Start on Windows Server](#) (p. 322)
- [Unable to Find Credentials on Windows Server](#) (p. 323)
- [CloudWatch Agent Files and Locations](#) (p. 323)
- [Logs Generated by the CloudWatch Agent](#) (p. 324)
- [Stopping and Restarting the CloudWatch Agent](#) (p. 324)

CloudWatch Agent Command Line Parameters

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


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

Installing the CloudWatch Agent Using Run Command Fails

To install the CloudWatch agent using Systems Manager Run Command, the SSM Agent on the target server must be version 2.2.93.0 or later. If your SSM Agent isn't the correct version, you might see errors that include the following messages:

```
no latest version found for package AmazonCloudWatchAgent on platform linux
```

```
failed to download installation package reliably
```

For information about updating your SSM Agent version, see [Installing and Configuring SSM Agent](#) in the *AWS Systems Manager User Guide*.

The CloudWatch Agent Won't Start

If the CloudWatch agent fails to start, there might be an issue in your configuration. Configuration information is logged in the `configuration-validation.log` file. This file is located in `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log` on Linux servers and in `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log` on servers running Windows Server.

Verify That the CloudWatch Agent Is Running

You can query the CloudWatch agent to find whether it's running or stopped. You can use AWS Systems Manager to do this remotely. You can also use the command line, but only to check the local server.

To query the status of the CloudWatch agent using Run Command

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

-or-

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

3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManageAgent**.
5. In the **Target** area, choose the instance to check.
6. In the **Action** list, choose **status**.
7. Keep **Optional Configuration Source** and **Optional Configuration Location** blank.
8. Choose **Run**.

If the agent is running, the output resembles the following.

```
{
  "status": "running",
  "starttime": "2017-12-12T18:41:18",
  "version": "1.73.4"
```

```
}
```

If the agent is stopped, the "status" field displays "stopped".

To query the status of the CloudWatch agent locally using the command line

- On a Linux server, enter the following:

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

On a server running Windows Server, enter the following in PowerShell as an administrator:

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2 -a status
```

Where Are the Metrics?

If the CloudWatch agent has been running but you can't find metrics collected by it in the AWS Management Console or the AWS CLI, confirm that you're using the correct namespace. By default, the namespace for metrics collected by the agent is `CWAgent`. You can customize this namespace using the `namespace` field in the `metrics` section of the agent configuration file. If you don't see the metrics that you expect, check the configuration file to confirm the namespace being used.

When you first download the CloudWatch agent package, the agent configuration file is `amazon-cloudwatch-agent.json`. This file is in the directory where you ran the configuration wizard, or you might have moved it to a different directory. If you use the configuration wizard, the agent configuration file output from the wizard is named `config.json`. For more information about the configuration file, including the `namespace` field, see [CloudWatch Agent Configuration File: Metrics Section \(p. 277\)](#).

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

If the agent doesn't start and the error message mentions an Amazon EC2 Region endpoint, you might have configured the agent to need access to the Amazon EC2 endpoint without granting that access.

For example, if you specify a value for the `append_dimensions` parameter in the agent configuration file that depends on Amazon EC2 metadata and you use proxies, you must make sure that the server can access the endpoint for Amazon EC2. For more information about these endpoints, see [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) in the *Amazon Web Services General Reference*.

The CloudWatch Agent Won't Start on Windows Server

On some Windows Server installations, the CloudWatch agent takes more than 30 seconds to start. Because Windows Server, by default, allows only 30 seconds for services to start, this causes the agent to fail with an error similar to the following:

```
Start-Service : Service 'Amazon CloudWatch Agent (AmazonCloudWatchAgent)' cannot be started
due to the following
error: Cannot start service AmazonCloudWatchAgent on computer '.'.
At C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1:113
char:12
```

```
+ $svc | Start-Service
+ ~~~~~
+ CategoryInfo          : OpenError:
(System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
ServiceCommandException
+ FullyQualifiedErrorId :
CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand
```

To fix this, increase the service timeout value. For more information, see [A service does not start, and events 7000 and 7011 are logged in the Windows event log](#).

Unable to Find Credentials on Windows Server

On Windows Server, if you have credentials in a location other than `$SystemDrive\Users\Administrator\.aws` on Windows Server 2008 R2 or Windows Server 2012, or `"$SystemDrive\Documents and Settings\Administrator\.aws` on Windows Server 2003, you can specify your own credential path by using the `shared_credential_file` option in `common.toml`.

If you don't have a credential file, you must create one. For more information, see [\(Optional\) Modify the Common Configuration for Proxy or Region Information \(p. 247\)](#).

CloudWatch Agent Files and Locations

The following table lists the files installed by and used with the CloudWatch agent, along with their locations on servers running Linux or Windows Server.

File	Linux Location	Windows Server Location
The control script that controls starting, stopping, and restarting the agent.	<code>/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl</code> or <code>/usr/bin/amazon-cloudwatch-agent-ctl</code>	<code>\$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1</code>
The log file the agent writes to. You might need to attach this when contacting AWS Support.	<code>/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log</code> or <code>/var/log/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.log</code>	<code>\$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log</code>
Agent configuration validation file.	<code>/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log</code> or <code>/var/log/amazon/amazon-cloudwatch-agent/configuration-validation.log</code>	<code>\$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log</code>
The JSON file used to configure the agent immediately after the wizard creates it. For more information, see Create the CloudWatch Agent Configuration File (p. 270) .	<code>/opt/aws/amazon-cloudwatch-agent/bin/config.json</code>	<code>\$Env:ProgramData\Amazon\AmazonCloudWatchAgent\config.json</code>

File	Linux Location	Windows Server Location
The JSON file used to configure the agent if this configuration file has been downloaded from Parameter Store.	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json or /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json	\$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json
TOML file used to specify Region and credential information to be used by the agent, overriding system defaults.	/opt/aws/amazon-cloudwatch-agent/etc/common-config.toml or /etc/amazon/amazon-cloudwatch-agent/common-config.toml	\$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.toml

Logs Generated by the CloudWatch Agent

The agent generates a log while it runs. This log includes troubleshooting information. This log is the `amazon-cloudwatch-agent.log` file. This file is located in `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` on Linux servers and in `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log` on servers running Windows Server.

You can configure the agent to log additional details in the `amazon-cloudwatch-agent.log` file. In the agent configuration file, in the `agent` section, set the `debug` field to `true`, then reconfigure and restart the CloudWatch agent. To disable the logging of this extra information, set the `debug` field to `false` reconfigure and restart the agent. For more information, see [Manually Create or Edit the CloudWatch Agent Configuration File](#) (p. 275).

Stopping and Restarting the CloudWatch Agent

You can manually stop the CloudWatch agent using either AWS Systems Manager or the command line.

To stop the CloudWatch agent using Run Command

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

-or-

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

3. Choose **Run command**.
4. In the **Command document** list, choose **AmazonCloudWatch-ManagedAgent**.
5. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
6. In the **Action** list, choose **stop**.
7. Keep **Optional Configuration Source** and **Optional Configuration Location** blank.
8. Choose **Run**.

To stop the CloudWatch agent locally using the command line

- On a Linux server, enter the following:

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

On a server running Windows Server, enter the following in PowerShell as an administrator:

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2  
-a stop
```

To restart the agent, follow the instructions in [Start the CloudWatch Agent \(p. 256\)](#).

Amazon CloudWatch Application Insights for .NET and SQL Server

Amazon CloudWatch Application Insights for .NET and SQL Server facilitates observability for .NET and SQL Server applications. It can help you set up the best monitors for your application resources to continuously analyze data for signs of problems with your applications. Application Insights, which is powered by [Sagemaker](#) and other AWS technologies, provides automated dashboards that show potential problems with monitored applications, helping you to quickly isolate ongoing issues with your applications and infrastructure. The enhanced visibility into the health of your applications that Application Insights provides can help you reduce your mean time to repair (MTTR) so that you don't have to pull in multiple teams and experts to troubleshoot your application issues.

When you add your applications to Amazon CloudWatch Application Insights for .NET and SQL Server, it scans the resources in the applications and recommends and configures metrics and logs on [CloudWatch](#) for application components. Application components may include SQL Server backend databases and Microsoft IIS/Web tiers. Application Insights analyzes metric patterns using historical data to detect anomalies, and continuously detects errors and exceptions from your application logs, including .NET, SQL Server error log, and IIS. It correlates these observations using a combination of classification algorithms and built-in rules. Then it automatically creates dashboards that show the relevant observations and problem severity information to help you prioritize your actions. For common problems in .NET and SQL application stacks, such as application latency, SQL Server failed backups, memory leaks, large HTTP requests, and aborted I/O operations, it provides additional insights that point to a possible root cause and steps for resolution. Built-in integration with [AWS Systems Manager OpsCenter](#) allows you to resolve issues by running the relevant Systems Manager Automation document.

Topics

- [What Is Amazon CloudWatch Application Insights for .NET and SQL Server? \(p. 326\)](#)
- [How Amazon CloudWatch Application Insights for .NET and SQL Server works \(p. 329\)](#)
- [Getting started with Amazon CloudWatch Application Insights for .NET and SQL Server \(p. 331\)](#)
- [Component configuration \(p. 344\)](#)
- [View and troubleshoot problems detected by Amazon CloudWatch Application Insights for .NET and SQL Server \(p. 353\)](#)
- [Logs and metrics supported by Amazon CloudWatch Application Insights for .NET and SQL Server \(p. 355\)](#)

What Is Amazon CloudWatch Application Insights for .NET and SQL Server?

CloudWatch Application Insights for .NET and SQL Server helps you monitor your .NET and SQL Server applications that use Amazon EC2 instances along with other [AWS application resources](#). It identifies and sets up key metrics, logs, and alarms across your application resources and technology stack (for example, your Microsoft SQL Server database, web (IIS) and application servers, OS, load balancers, and queues). It continuously monitors metrics and logs to detect and correlate anomalies and errors. When errors and anomalies are detected, Application Insights generates [CloudWatch Events](#) that you can use to set up notifications or take actions. To aid with troubleshooting, it creates automated dashboards for detected problems, which include correlated metric anomalies and log errors, along with additional insights to point you to a potential root cause. The automated dashboards help you to take remedial actions to keep your applications healthy and to prevent impact to the end users of your application. It also creates OpsItems so that you can resolve problems using [AWS Systems Manager OpsCenter](#).

Contents

- [Features \(p. 327\)](#)
- [Concepts \(p. 327\)](#)
- [Pricing \(p. 328\)](#)
- [Related AWS services \(p. 328\)](#)
- [Supported application components \(p. 329\)](#)
- [Supported technology stack \(p. 329\)](#)

Features

Application Insights provides the following features.

Automatic set up of monitors for application resources

CloudWatch Application Insights for .NET and SQL Server reduces the time it takes to set up monitoring for your applications. It does this by scanning your application resources, providing a customizable list of recommended metrics and logs, and setting them up on CloudWatch to provide necessary visibility into your application resources, such as Amazon EC2 and Elastic Load Balancers (ELB). It also sets up dynamic alarms on monitored metrics. The alarms are automatically updated based on anomalies detected in the previous two weeks.

Problem detection and notification

CloudWatch Application Insights for .NET and SQL Server detects signs of potential problems with your application, such as metric anomalies and log errors. It correlates these observations to surface potential problems with your application. It then generates CloudWatch Events, [which can be configured to receive notifications or take actions \(p. 343\)](#). This eliminates the need for you to create individual alarms on metrics or log errors.

Troubleshooting

CloudWatch Application Insights for .NET and SQL Server creates CloudWatch automatic dashboards for problems that are detected. The dashboards show details about the problem, including the associated metric anomalies and log errors to help you with troubleshooting. They also provide additional insights that point to potential root causes of the anomalies and errors.

Concepts

The following concepts are important for understanding how Application Insights monitors your application.

Component

An auto-grouped, standalone, or custom grouping of similar resources that make up an application. We recommend grouping similar resources into custom components for better monitoring.

Observation

An individual event (metric anomaly, log error, or exception) that is detected with an application or application resource.

Problem

Problems are detected by correlating, classifying, and grouping related observations.

For definitions of other key concepts for CloudWatch Application Insights for .NET and SQL Server, see [Amazon CloudWatch Concepts](#).

Pricing

CloudWatch Application Insights for .NET and SQL Server sets up recommended metrics and logs for selected application resources using CloudWatch Metrics, Logs, and CloudWatch Events for notifications on detected problems. These features are charged to your AWS account according to [CloudWatch pricing](#). For the detected problems, it creates [CloudWatch Events](#) and [automatic dashboards](#). You are not charged for setup assistance, monitoring data analysis, or problem detection.

Related AWS services

The following services are used along with CloudWatch Application Insights for .NET and SQL Server:

- **Amazon CloudWatch** provides system-wide visibility into resource utilization, application performance, and operational health. It collects and tracks metrics, sends alarm notifications, automatically updates resources that you are monitoring based on the rules that you define, and allows you to monitor your own custom metrics. CloudWatch Application Insights for .NET and SQL Server is initiated through CloudWatch—specifically, within the CloudWatch default operational dashboards. For more information, see the [Amazon CloudWatch User Guide](#).
- **AWS Resource Groups** help you to organize the resources that make up your application. With Resource Groups, you can manage and automate tasks on a large number of resources at one time. Only one Resource Group can be registered for a single application. For more information, see the [AWS Resource Groups User Guide](#).
- **IAM** is a web service that helps you to securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication), and to control the resources they can use and how they can use them (authorization). For more information, see [Authentication and Access Control for Amazon CloudWatch](#).
- **Amazon EC2** provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, to configure security and networking, and to manage storage. You can scale up or down to handle changes in requirements or spikes in popularity, which reduces your need to forecast traffic. For more information, see the [Amazon EC2 User Guide for Linux Instances](#) or [Amazon EC2 Guide for Windows Instances](#).
- **Elastic Load Balancing** distributes incoming applications or network traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in multiple Availability Zones. For more information, see the [Elastic Load Balancing User Guide](#).
- **Amazon EC2 Auto Scaling** helps ensure that you have the correct number of EC2 instances available to handle the load for your application. For more information, see the [Amazon EC2 Auto Scaling User Guide](#).
- **Amazon SQS** offers a secure, durable, and available hosted queue that allows you to integrate and decouple distributed software systems and components. For more information, see the [Amazon SQS User Guide](#).
- **AWS Systems Manager OpsCenter** aggregates and standardizes OpsItems across services while providing contextual investigation data about each OpsItem, related OpsItems, and related resources. OpsCenter also provides Systems Manager Automation documents (runbooks) that you can use to quickly resolve issues. You can specify searchable, custom data for each OpsItem. You can also view automatically-generated summary reports about OpsItems by status and source. For more information, see the [AWS Systems Manager User Guide](#).
- **AWS Lambda** lets you build serverless applications composed of functions that are triggered by events and automatically deploy them using CodePipeline and AWS CodeBuild. For more information, see [AWS Lambda Applications](#).
- **Amazon DynamoDB** is a fully managed NoSQL database service that lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

Supported application components

CloudWatch Application Insights for .NET and SQL Server scans your Resource Group—in this case, a group of resources used by a .NET application—to identify application components. Components can be standalone, auto-grouped (such as instances in an Auto Scaling group or behind a load balancer), or custom (by grouping together individual EC2 instances). The following components are supported by CloudWatch Application Insights for .NET and SQL Server:

- Amazon EC2
- Amazon RDS
- Elastic Load Balancing: Application Load Balancer and Classic Load Balancer (all target instances of these load balancers are identified and configured).
- Amazon EC2 Auto Scaling groups: AWS Auto Scaling (Auto Scaling groups are dynamically configured for all target instances; if your application scales up, CloudWatch Application Insights for .NET and SQL Server automatically configure the new instances). Auto Scaling groups are not supported for CloudFormation-based Resource Groups.
- AWS Lambda
- Amazon Simple Queue Service (Amazon SQS)
- Amazon DynamoDB table
- Amazon S3 bucket metrics

Any other component type resources are ignored by CloudWatch Application Insights for .NET and SQL Server. If a component type that is supported does not appear in your Application Insights application, the component may already be registered and managed by another application you own that is monitored by Application Insights.

Supported technology stack

CloudWatch Application Insights for .NET and SQL Server supports:

- Front-end: Microsoft Internet Information Services (IIS) Web Server
- Worker-tier: .NET Framework
- Databases:
 - Microsoft SQL Server running on RDS or EC2
 - Amazon DynamoDB table

If none of the technology stacks listed above apply to your application resources, you can monitor your application stack by choosing **Custom** from the application tier dropdown menu on the **Manage monitoring** page.

How Amazon CloudWatch Application Insights for .NET and SQL Server works

This section contains information about how CloudWatch Application Insights for .NET and SQL Server works, including:

Topics

- [How Application Insights monitors applications \(p. 330\)](#)
- [Data retention \(p. 330\)](#)

- [Quotas \(p. 330\)](#)

How Application Insights monitors applications

Application Insights monitors .NET and SQL Server applications as follows.

Application discovery and configuration

The first time an application is added to CloudWatch Application Insights for .NET and SQL Server it scans the application components to recommend key metrics, logs, and other data sources to monitor for your application. You can then configure your application based on these recommendations.

Data preprocessing

CloudWatch Application Insights for .NET and SQL Server continuously analyzes the data sources being monitored across the application resources to discover metric anomalies and log errors (observations).

Intelligent problem detection

The CloudWatch Application Insights engine detects problems in your application by correlating observations using classification algorithms and built-in rules. To assist in troubleshooting, it creates automated CloudWatch dashboards, which include contextual information about the problems.

Alert and action

When CloudWatch Application Insights detects a problem with your application, it generates CloudWatch Events to notify you of the problem. See [Set up notifications and actions for detected problems \(p. 343\)](#) for more information about how to set up these Events.

Example scenario

You have an ASP .NET application that is backed by a SQL Server database. Suddenly, your database begins to malfunction because of high memory pressure. This leads to application performance degradation and possibly HTTP 500 errors in your web servers and load balancer.

With CloudWatch Application Insights for .NET and SQL Server and its intelligent analytics, you can identify the application layer that is causing the problem by checking the dynamically created dashboard that shows the related metrics and log file snippets. In this case, the problem might be at the SQL database layer.

Data retention

CloudWatch Application Insights for .NET and SQL Server retains problems for 55 days and observations for 60 days.

Quotas

The following table provides the default quotas for CloudWatch Application Insights for .NET and SQL Server. Unless otherwise noted, each quota is per AWS Region. Please contact [AWS Support](#) to request an increase in your service quota. Many services contain quotas that cannot be changed. For more information about the quotas for a specific service, see the documentation for that service.

Resource	Default quota
API requests	All API actions are throttled to 5 TPS
Applications	10 per account

Resource	Default quota
Log Streams	5 per resource
Observations per problem	20 per dashboard 40 per DescribeProblemObservations action
Metrics	10 per resource
Resources	30 per application

Getting started with Amazon CloudWatch Application Insights for .NET and SQL Server

To get started on CloudWatch Application Insights for .NET and SQL Server, verify that you have met the prerequisites outlined below and have created an IAM policy. Then, you can get started using the console link to enable CloudWatch Application Insights for .NET and SQL Server. To configure your application resources, follow the steps under [Setting up your application for monitoring \(p. 333\)](#).

Contents

- [Accessing CloudWatch Application Insights for .NET and SQL Server \(p. 331\)](#)
- [Prerequisites \(p. 331\)](#)
- [IAM policy \(p. 332\)](#)
- [Setting up your application for monitoring \(p. 333\)](#)

Accessing CloudWatch Application Insights for .NET and SQL Server

If you have access to CloudWatch Application Insights for .NET and SQL Server, you can manage it through one of the following interfaces:

- **CloudWatch console:** To add monitors for your application, choose **Settings** in the left navigation pane of the [CloudWatch console](#). From the **Settings** page, choose **Application Insights for .NET and SQL Server**. After your application is configured, you can use the [CloudWatch console](#) to view and analyze problems that are detected.
- **AWS Command Line Interface (AWS CLI):** You can use the AWS CLI to access AWS API operations. For more information, see [Installing the AWS Command Line Interface](#) in the *AWS Command Line Interface User Guide*. For the API reference, see the [Amazon CloudWatch Application Insights for .NET and SQL Server API Reference](#).

Prerequisites

You must complete the following prerequisites to configure an application with CloudWatch Application Insights:

- **AWS Systems Manager enablement:** You must install Systems Manager Agent (SSM Agent), and your instances must be SSM enabled. For steps on how to install the SSM Agent, see [Setting Up AWS Systems Manager](#).

- **EC2 instance role:** You must attach the `AmazonSSMManagedInstanceCore` role to enable Systems Manager (see [Using Identity-based Policies \(IAM Policies\) for AWS Systems Manager](#)) and the `CloudWatchAgentServerPolicy` to enable instance metrics and logs to be emitted through CloudWatch. See [Create IAM Roles and Users for Use With CloudWatch Agent](#) for more information.
- **AWS Resource Groups:** To onboard your .NET and SQL Server applications to CloudWatch Application Insights for .NET and SQL Server, you must create a resource group that includes all associated AWS resources used by your application stack. This includes application load balancers, EC2 instances running IIS and web front-end, .NET worker tiers, and your SQL Server database. CloudWatch Application Insights for .NET and SQL Server automatically includes Auto Scaling groups using the same tags or CloudFormation stacks as your Resource Group, because Auto Scaling groups are not currently supported by Resource Groups. For more information, see [Getting Started with AWS Resource Groups](#).
- **IAM permissions:** For non-Admin users, you must create an AWS Identity and Access Management (IAM) Policy and attach it to your user identity. See [IAM policy \(p. 332\)](#).
- **Service-linked role:** CloudWatch Application Insights for .NET and SQL Server uses AWS Identity and Access Management (IAM) service-linked roles. You don't need to manually create a service-linked role. Instead, it is created for you when you create your first CloudWatch Application Insights for .NET and SQL Server application in the AWS Management Console. For more information, see [Using Service-Linked Roles for CloudWatch Application Insights for .NET and SQL Server \(p. 436\)](#).
- **Performance Counter metrics support for EC2 Windows instances:** To monitor Performance Counter metrics on your EC2 Windows instances, Performance Counters must be installed on the instances. For Performance Counter metrics and corresponding Performance Counter set names, see [Performance Counter metrics \(p. 374\)](#). For more information about Performance Counters, see [Performance Counters](#).

IAM policy

To use CloudWatch Application Insights for .NET and SQL Server, you must create an [Identity and Access Management \(IAM\) policy](#) and attach it to your IAM user identity. The IAM policy defines the user permissions.

To create an IAM policy using the console

To create an IAM policy using the IAM console, follow these steps.

1. Go to the [IAM console](#). In the left navigation pane, select **Policies**.
2. At the top of the page, select **Create policy**.
3. Select the **JSON** tab.
4. Copy and paste the following JSON document under the **JSON** tab.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "applicationinsights:*",
        "iam:CreateServiceLinkedRole",
        "iam:ListRoles"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

5. Select **Review Policy**.

6. Enter a **Name** for the policy, for example, "AppInsightsPolicy." Optionally, enter a **Description**.
7. Select **Create Policy**.
8. Select **Users** from the left navigation pane.
9. Select the **User name** of the user to which you would like to attach the policy.
10. Select **Add permissions**.
11. Select **Attach existing policies directly**.
12. Search for the policy that you just created, and select the check box to the left of the policy name.
13. Select **Next: Review**.
14. Make sure that the correct policy is listed, and select **Add permissions**.
15. Make sure that you log in with the user associated with the policy that you just created when you use CloudWatch Application Insights for .NET and SQL Server.

To create an IAM policy using the AWS CLI

To create an IAM policy using the AWS CLI, run the [create-policy](#) operation from the command line using the JSON document above as a file in your current folder.

To create an IAM policy using AWS Tools for Windows PowerShell

To create an IAM policy using the AWS Tools for Windows PowerShell, run the [New-IAMPolicy](#) cmdlet using the JSON document above as a file in your current folder.

Setting up your application for monitoring

This section provides steps for setting up, configuring, and managing your CloudWatch Application Insights for .NET and SQL Server application using the console, the AWS CLI, and AWS Tools for Windows PowerShell.

Topics

- [Add and configure an application with the CloudWatch console \(p. 333\)](#)
- [Disable an application with the console \(p. 335\)](#)
- [Disable monitoring for an application component with the console \(p. 336\)](#)
- [Delete an Application with the console \(p. 336\)](#)
- [Add and Manage an application with Application Insights using the command line \(p. 336\)](#)
- [Manage and update monitoring using the command line \(p. 339\)](#)
- [Configure monitoring for MySQL RDS \(p. 342\)](#)
- [Configure monitoring for MySQL EC2 \(p. 343\)](#)
- [Set up notifications and actions for detected problems \(p. 343\)](#)

Add and configure an application with the CloudWatch console

Add and configure an application from the CloudWatch console

To get started with CloudWatch Application Insights for .NET and SQL Server from the CloudWatch console, follow these steps.

1. **Start.** Open the [CloudWatch console landing page](#). From the left navigation pane, choose **Settings**. From the **Settings** page, choose **Application Insights for .NET and SQL Server > View applications to get started**.
2. **Add an Application.** To set up monitoring for your .NET and SQL Server application, on the CloudWatch Application Insights for .NET and SQL Server page, select **Add an application**. This page

shows the list of applications that are monitored with CloudWatch Application Insights for .NET and SQL Server, along with their monitoring status. After you select **Add an application**, you will be taken to the **Add an application** page.

3. **Select Resource Group.** On the **Add an application** page, to add an application to CloudWatch Application Insights for .NET and SQL Server, choose an AWS Resource Group from the dropdown list that contains your application resources. These resources include front-end servers, load balancers, auto scaling groups, and database servers.

An **ARN** will be generated for the application in the following format:

```
arn:partition:applicationinsights:region:account-id:application/resource-group/resource-group-name
```

For example:

```
arn:aws:applicationinsights:us-east-1:123456789012:application/resource-group/my-resource-group
```

CloudWatch Application Insights for .NET and SQL Server supports both tag-based and CloudFormation-based Resource Groups (with the exception of Auto Scaling groups). For more information, see [Working with Tag Editor](#).

If you have not created a Resource Group for your .NET application, you can create one. For more information, see the [AWS Resource Groups User Guide](#).

4. **Add Monitoring Details.** After you add an application, you are taken to the **Monitoring Details** page, which lists the application components, resources in those components, and their monitoring status. Components are auto-grouped, standalone, or custom groupings of similar resources that make up an application. By default, CloudWatch Application Insights for .NET and SQL Server groups instances that are in Auto Scaling groups, and instances that are behind your Elastic Load Balancers. On this page, you can configure custom components and can manage monitoring for each application component. For supported components, see [Supported application components](#) (p. 329).
5. **Configure components.** After selecting a Resource Group, you are prompted to configure [components](#). We recommend grouping similar resources, such as .NET web server instances, into custom components for easier onboarding and better monitoring and insights. By default, CloudWatch Application Insights for .NET and SQL Server groups instances that are in Auto Scaling groups, and instances that are behind your Elastic Load Balancers. For supported components, see [Supported application components](#) (p. 329).

Under **Application components**, for each component for which you want to set up monitors, select the component and select **Manage Monitoring**.

6. **Enable Monitors.** To set up monitoring for an application component, select the components that you want to monitor and then choose **Manage Monitoring**. Select the **Enable Monitoring** check box. When you select the check box, the dropdown populates with the relevant application tiers. Choose the application tier for the selected component. The tiers indicate the part of the application stack running on the selected resources.

Based on your tier selection, CloudWatch Application Insights for .NET and SQL Server makes recommendations for logs to monitor for the selected component. This recommendation can be customized according to your needs.

For application-specific logs, including for Microsoft SQL Server Error logs and IIS logs, verify the default log path (if any) or enter the correct log location in your EC2 instance.

You can also choose to add Windows Event Logs, including Windows Logs and Applications and Services Logs. To do this, enter the types of events you want to store and analyze. Then, specify all

of the event levels (critical, error, warning, informational, or verbose) that you want to store in your CloudWatch account.

You can add a log group for storing and grouping each of these logs on your CloudWatch account, which also facilitates searches.

CloudWatch Application Insights for .NET and SQL Server also sets up relevant metrics for your application resources. They are monitored for approximately two weeks to identify the appropriate metrics thresholds. If you have created the metrics in the past, CloudWatch Application Insights for .NET and SQL Server pulls historical data for the last two weeks to identify the thresholds and to set the alarms accordingly. For newly created metrics, it may take up to three days before alarms are created. You can also monitor your application resources using the CloudWatch alarms you created in your account.

7. **Save monitors.** When you are finished selecting and customizing logs and metrics, select **Save** to set up monitors for the selected component. When you select **Save**, Application Insights sets up the CloudWatch Agent configuration files for all of the instances in your application based on the recommended metrics and your selection of logs. It can take up to an hour for this process to complete.

CloudWatch Application Insights for .NET and SQL Server also sets up CloudWatch alarms for selected metrics in the component. The alarms are dynamically updated by monitoring historical metric patterns from the past two weeks.

When you select **Cancel**, Application Insights only deletes your current selections.

When you create a new application with CloudWatch Application Insights for .NET and SQL Server, the service-linked role is created for you. To delete the service-linked role, you must first delete all of your applications on CloudWatch Applications Insights for .NET and SQL Server and then manually delete the role. For more information, see [Using Service-Linked Roles for CloudWatch Application Insights for .NET and SQL Server \(p. 436\)](#).

CloudWatch Application Insights for .NET and SQL Server is now set to monitor metrics and logs for your application. It may take up to two weeks for the system to generate meaningful insights.

*If your Resource Group is already configured and you want to save your configuration but don't want CloudWatch Application Insights for .NET and SQL Server to monitor your application, you can disable CloudWatch Application Insights for .NET and SQL Server. You can also delete your configuration.

8. **Add AWS Systems Manager OpsCenter Integration.** To view and get notified when problems are detected for selected applications, select the **Integrate with AWS OpsCenter** check box on the **Monitoring Details** page. To track the operations that are taken to resolve operational work items (OpsItems) that are related to your AWS resources, provide the SNS topic ARN.
9. **View monitoring (optional).** After your application has been set up for monitoring, you can view and troubleshoot detected problems and insights in the default overview page of the CloudWatch console. You can view detected problems, alarms, and dashboards by selecting **View Insights** from the Application Insights landing page, or on the CloudWatch landing page.

Disable an application with the console

To disable an application, from the CloudWatch dashboard, under **Settings**, select the application that you want to disable. Under **Actions**, choose **Disable**. When you disable an application, monitoring is disabled, but Application Insights stores the saved monitors for application components.

Disable monitoring for an application component with the console

To disable monitoring for an application component, from the Application Details page select the component for which you want to disable monitoring. Choose **Manage Monitors**, and then uncheck the **Enable Monitoring** check box.

Delete an Application with the console

To delete an application, from the CloudWatch dashboard, under **Settings**, select the application that you want to delete. Under **Actions**, choose **Delete**. This deletes monitoring and deletes all of the saved monitors for application components. The application resources are not deleted.

Add and Manage an application with Application Insights using the command line

You can add, get information about, manage, and configure your Application Insights application using the command line.

Topics

- [Add an application \(p. 336\)](#)
- [Describe an application \(p. 336\)](#)
- [List components in an application \(p. 337\)](#)
- [Describe a component \(p. 337\)](#)
- [Group similar resources into a custom component \(p. 337\)](#)
- [Ungroup a custom component \(p. 338\)](#)
- [Update an application \(p. 338\)](#)
- [Update a custom component \(p. 339\)](#)

Add an application

Add an application using the AWS CLI

To use the AWS CLI to add an application for your resource group called `my-resource-group`, with OpsCenter enabled to deliver the created opsItem to the SNS topic ARN `arn:aws:sns:us-east-1:123456789012:MyTopic`, use the following command.

```
aws application-insights create-application --resource-group-name my-resource-group --ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Add an application using AWS Tools for Windows PowerShell

To use AWS Tools for Windows PowerShell to add an application for your resource group called `my-resource-group` with OpsCenter enabled to deliver the created opsItem to the SNS topic ARN `arn:aws:sns:us-east-1:123456789012:MyTopic`, use the following command.

```
New-CWAIAApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Describe an application

Describe an application using the AWS CLI

To use the AWS CLI to describe an application created on a resource group called `my-resource-group`, use the following command.

```
aws application-insights describe-application --resource-group-name my-resource-group
```

Describe an application using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to describe an application created on a resource group called `my-resource-group`, use the following command.

```
Get-CWAIAApplication -ResourceGroupName my-resource-group
```

List components in an application

List components in an application using the AWS CLI

To use the AWS CLI to list the components created on a resource group called `my-resource-group`, use the following command.

```
aws application-insights list-components --resource-group-name my-resource-group
```

List components in an application using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to list the components created on a resource group called `my-resource-group`, use the following command.

```
Get-CWAIDComponentList -ResourceGroupName my-resource-group
```

Describe a component

Describe a component using the AWS CLI

You can use the following AWS CLI command to describe a component called `my-component` that belongs to an application created on a resource group called `my-resource-group`.

```
aws application-insights describe-component --resource-group-name my-resource-group --component-name my-component
```

Describe a component using AWS Tools for Windows PowerShell

You can use the following AWS Tools for Windows PowerShell command to describe a component called `my-component` that belongs to an application created on a resource group called `my-resource-group`.

```
Get-CWAIDComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

Group similar resources into a custom component

We recommend grouping similar resources, such as .NET web server instances, into custom components for easier onboarding and better monitoring and insights. Currently, CloudWatch Application Insights for .NET and SQL Server supports custom groups for EC2 instances.

To group resources into a custom component using the AWS CLI

To use the AWS CLI to group three instances (`arn:aws:ec2:us-east-1:123456789012:instance/i-11111`, `arn:aws:ec2:us-east-1:123456789012:instance/i-22222`, and `arn:aws:ec2:us-`

east-1:123456789012:instance/i-33333) together into a custom component called `my-component` for an application created for the resource group called `my-resource-group`, use the following command.

```
aws application-insights create-component --resource-group-name my-resource-group --
component-name my-component --resource-list arn:aws:ec2:us-east-1:123456789012:instance/
i-11111 arn:aws:ec2:us-east-1:123456789012:instance/i-22222 arn:aws:ec2:us-
east-1:123456789012:instance/i-33333
```

To group resources into a custom component using AWS Tools for Windows PowerShell

To use AWS Tools for Windows PowerShell to group three instances (`arn:aws:ec2:us-east-1:123456789012:instance/i-11111`, `arn:aws:ec2:us-east-1:123456789012:instance/i-22222`, and `arn:aws:ec2:us-east-1:123456789012:instance/i-33333`) together into a custom component called `my-component`, for an application created for the resource group called `my-resource-group`, use the following command.

```
New-CWAComponent -ResourceGroupName my-resource-group -ComponentName my-component
-ResourceList arn:aws:ec2:us-east-1:123456789012:instance/i-11111,arn:aws:ec2:us-
east-1:123456789012:instance/i-22222,arn:aws:ec2:us-east-1:123456789012:instance/i-33333
```

Ungroup a custom component

To ungroup a custom component using the AWS CLI

To use the AWS CLI to ungroup a custom component named `my-component` in an application created on the resource group, `my-resource-group`, use the following command.

```
aws application-insights delete-component --resource-group-name my-resource-group --
component-name my-new-component
```

To ungroup a custom component using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to ungroup a custom component named `my-component` in an application created on the resource group, `my-resource-group`, use the following command.

```
Remove-CWAComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

Update an application

Update an application using the AWS CLI

You can use the AWS CLI to update an application to generate AWS Systems Manager OpsCenter OpsItems for problems detected with the application, and to associate the created OpsItems to the SNS topic `arn:aws:sns:us-east-1:123456789012:MyTopic`, using the following command.

```
aws application-insights update-application --resource-group-name my-resource-group --ops-
center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Update an application using AWS Tools for Windows PowerShell

You can use the AWS Tools for Windows PowerShell to update an application to generate AWS Systems Manager OpsCenter OpsItems for problems detected with the application, and to associate the created OpsItems to the SNS topic `arn:aws:sns:us-east-1:123456789012:MyTopic`, using the following command.

```
Update-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -  
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

Update a custom component

Update a custom component using the AWS CLI

You can use the AWS CLI to update a custom component called `my-component` with a new component name, `my-new-component`, and an updated group of instances, by using the following command.

```
aws application-insights update-component --resource-group-name my-resource-  
group --component-name my-component --new-component-name my-new-  
component --resource-list arn:aws:ec2:us-east-1:123456789012:instance/i-44444 arn:aws:ec2:us-  
east-1:123456789012:instance/i-55555
```

Update a custom component using AWS Tools for Windows PowerShell

You can use the AWS Tools for Windows PowerShell to update a custom component called `my-component` with a new component name, `my-new-component`, and an updated group of instances, by using the following command.

```
Update-CWAComponent -ComponentName my-component -NewComponentName my-new-  
component -ResourceGroupName my-resource-group -ResourceList arn:aws:ec2:us-  
east-1:123456789012:instance/i-44444,arn:aws:ec2:us-east-1:123456789012:instance/i-55555
```

Manage and update monitoring using the command line

You can manage and update monitoring for your Application Insights application using the command line.

Topics

- [List problems with your application \(p. 339\)](#)
- [Describe an application problem \(p. 340\)](#)
- [Describe the anomalies or errors associated with a problem \(p. 340\)](#)
- [Describe an anomaly or error with the application \(p. 340\)](#)
- [Describe the monitoring configurations of a component \(p. 341\)](#)
- [Describe the recommended monitoring configuration of a component \(p. 341\)](#)
- [Update the monitoring configurations for a component \(p. 341\)](#)
- [Remove a specified resource group from Application Insights monitoring \(p. 341\)](#)

List problems with your application

List problems with your application using the AWS CLI

To use the AWS CLI to list problems with your application detected between 1,000 and 10,000 milliseconds since Unix Epoch for an application created on a resource group called `my-resource-group`, use the following command.

```
aws application-insights list-problems --resource-group-name my-resource-group --start-  
time 1000 --end-time 10000
```

List problems with your application using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to list problems with your application detected between 1,000 and 10,000 milliseconds since Unix Epoch for an application created on a resource group called `my-resource-group`, use the following command.

```
$startDate = "8/6/2019 3:33:00"  
$endDate = "8/6/2019 3:34:00"  
Get-CWAIProblemList -ResourceGroupName my-resource-group -StartTime $startDate -  
EndTime $endDate
```

Describe an application problem

Describe an application problem using the AWS CLI

To use the AWS CLI to describe a problem with problem id `p-1234567890`, use the following command.

```
aws application-insights describe-problem --problem-id p-1234567890
```

Describe an application problem using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to describe a problem with problem id `p-1234567890`, use the following command.

```
Get-CWAIProblem -ProblemId p-1234567890
```

Describe the anomalies or errors associated with a problem

Describe the anomalies or errors associated with a problem using the AWS CLI

To use the AWS CLI to describe the anomalies or errors associated with a problem with problem id `p-1234567890`, use the following command.

```
aws application-insights describe-problem-observations --problem-id p-1234567890
```

Describe the anomalies or errors associated with a problem using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to describe the anomalies or errors associated with a problem with problem id `p-1234567890`, use the following command.

```
Get-CWAIProblemObservation -ProblemId p-1234567890
```

Describe an anomaly or error with the application

Describe an anomaly or error with the application using the AWS CLI

To use the AWS CLI to describe an anomaly or error with the application with the observation id `o-1234567890`, use the following command.

```
aws application-insights describe-observation --observation-id o-1234567890
```

Describe an anomaly or error with the application using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to describe an anomaly or error with the application with the observation id `o-1234567890`, use the following command.

```
Get-CWAIObservation -ObservationId o-1234567890
```

Describe the monitoring configurations of a component

Describe the monitoring configurations of a component using the AWS CLI

To use the AWS CLI to describe the monitoring configuration of a component called `my-component` in an application created on the resource group `my-resource-group`, use the following command.

```
aws application-insights describe-component-configuration --resource-group-name my-resource-group --component-name my-component
```

Describe the monitoring configurations of a component using AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to describe the monitoring configuration of a component called `my-component`, in an application created on the resource group `my-resource-group`, use the following command.

```
Get-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-resource-group
```

For more information about component configuration and for example JSON files, see [Component configuration](#) (p. 344).

Describe the recommended monitoring configuration of a component

Describe the recommended monitoring configuration of a component using the AWS CLI

When the component is part of a .NET Worker application, you can use the AWS CLI to describe the recommended monitoring configuration of a component called `my-component` in an application created on the resource group `my-resource-group`, by using the following command.

```
aws application-insights describe-component-configuration-recommendation --resource-group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER
```

Describe the recommended monitoring configuration of a component using AWS Tools for Windows PowerShell

When the component is part of a .NET Worker application, you can use the AWS Tools for Windows PowerShell to describe the recommended monitoring configuration of a component called `my-component` in an application created on the resource group `my-resource-group`, by using the following command.

```
Get-CWAIComponentConfigurationRecommendation -ComponentName my-component -ResourceGroupName my-resource-group -Tier DOT_NET_WORKER
```

For more information about component configuration and for example JSON files, see [Component configuration](#) (p. 344).

Update the monitoring configurations for a component

Update the monitoring configurations for a component using the AWS CLI

To use the AWS CLI to update the component called `my-component` in an application created on the resource group called `my-resource-group`, use the following command. The command includes these actions:

1. Enable monitoring for the component.
2. Set the tier of the component to `.NET Worker`.

3. Update the JSON configuration of the component to read from the local file `configuration.txt`.

```
aws application-insights update-component-configuration --resource-group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER --monitor --component-configuration "file://configuration.txt"
```

Update the monitoring configurations for a component using the AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to update the component called `my-component` in an application created on the resource group called `my-resource-group`, use the following command. The command includes these actions:

1. Enable monitoring for the component.
2. Set the tier of the component to `.NET Worker`.
3. Update the JSON configuration of the component to read from the local file `configuration.txt`.

```
[string]$config = Get-Content -Path configuration.txt  
Update-CWAIDComponentConfiguration -ComponentName my-component -ResourceGroupName my-resource-group -Tier DOT_NET_WORKER -Monitor 1 -ComponentConfiguration $config
```

For more information about component configuration and for example JSON files, see [Component configuration \(p. 344\)](#).

Remove a specified resource group from Application Insights monitoring

Remove a specified resource group from Application Insights monitoring using the AWS CLI

To use the AWS CLI to remove an application created on the resource group called `my-resource-group` from monitoring, use the following command.

```
aws application-insights delete-application --resource-group-name my-resource-group
```

Remove a specified resource group from Application Insights monitoring using the AWS Tools for Windows PowerShell

To use the AWS Tools for Windows PowerShell to remove an application created on the resource group called `my-resource-group` from monitoring, use the following command.

```
Remove-CWAIDApplication -ResourceGroupName my-resource-group
```

Configure monitoring for MySQL RDS

1. Create an application for the resource group with the RDS MySQL database instance.

```
aws application-insights create-application --region <REGION> --resource-group-name <RESOURCE_GROUP_NAME>
```

2. The error log is enabled by default. The slow query log can be enabled using data parameter groups. For more information, see [Accessing the MySQL Slow Query and General Logs](#).
 - `set slow_query_log = 1`
 - `set log_output = FILE`
3. Export the logs to be monitored to CloudWatch logs. For more information, see [Publishing MySQL Logs to CloudWatch Logs](#).

4. Configure the MySQL RDS component.

```
aws application-insights update-component-configuration --resource-group-name
"<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
--monitor --tier DEFAULT --monitor --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logType\":
\"MySQL\",\"monitor\":true},{\"logType\":\"MySQL_SLOW_QUERY\",\"monitor\":false}]}"
```

Configure monitoring for MySQL EC2

1. Create an application for the resource group with the SQL HA EC2 instances.

```
aws application-insights create-application --region <REGION> --resource-group-name
<RESOURCE_GROUP_NAME>
```

2. The error log is enabled by default. The slow query log can be enabled using data parameter groups. For more information, see [Accessing the MySQL Slow Query and General Logs](#).

- set slow_query_log = 1
- set log_output = FILE

3. Configure the MySQL EC2 component.

```
aws application-insights update-component-configuration --resource-group-name
"<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
--monitor --tier MYSQL --monitor --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logGroupName
\":\"<UNIQUE_LOG_GROUP_NAME>\",\"logPath\":\"C:\\\\ProgramData\\\\MySQL\\\\MySQL Server
*\\\\Data\\\\<FILE_NAME>.err\",\"logType\":\"MYSQL\",\"monitor\":true,\"encoding\":
\"utf-8\"}]}"
```

Set up notifications and actions for detected problems

For each application that is added to CloudWatch Application Insights for .NET and SQL Server, a CloudWatch Event is published for the following events:

- **Problem Creation.** Emitted when CloudWatch Application Insights for .NET and SQL Server detects a new problem.
 - Detail Type: **"Application Insights Problem Detected"**
 - Detail:
 - problemId: The detected problem ID.
 - region: The AWS Region where the problem was created.
 - resourceGroupName: The Resource Group for the registered application for which the problem was detected.
 - status: The status of the problem.
 - severity: The severity of the problem.
 - problemUrl: The console URL for the problem.
- **Problem Update.** Emitted when the problem is updated with a new observation or when an existing observation is updated and the problem is subsequently updated; updates include a resolution or closure of the problem.
 - Detail Type: **"Application Insights Problem Updated"**
 - Detail:

- `problemId`: The created problem ID.
- `region`: The AWS Region where the problem was created.
- `resourceGroupName`: The Resource Group for the registered application for which the problem was detected.
- `status`: The status of the problem.
- `severity`: The severity of the problem.
- `problemUrl`: The console URL for the problem.

How to receive notification for problem events generated by an application

From the CloudWatch console, select **Rules** under **Events** in the left navigation pane. From the **Rules** page, select **Create rule**. Choose **CloudWatch Application Insights** from the **Service Name** dropdown list and choose the **Event Type**. Then, choose **Add target** and select the target and parameters, for example, an **SNS topic** or **Lambda function**.

Actions through AWS Systems Manager. CloudWatch Application Insights for .NET and SQL Server provides built-in integration with Systems Manager OpsCenter. If you choose to use this integration for your application, an OpsItem is created on the OpsCenter console for every problem detected with the application. From the OpsCenter console, you can view summarized information about the problem detected by CloudWatch Application Insights and pick a Systems Manager Automation runbook to take remedial actions or further identify Windows processes that are causing resource issues in your application.

Component configuration

A component configuration is a text file in JSON format that describes the configuration settings of the component. This section provides examples of a component configuration and its sections.

JSON

The following example shows a template fragment in JSON format.

```
{
  "alarmMetrics" : [
    list of alarm metrics
  ],
  "logs" : [
    list of logs
  ],
  "instances" : [
    component nested instances configuration
  ],
  "windowsEvents" : [
    list of windows events channels configurations
  ],
  "alarms" : [
    list of CloudWatch alarms
  ]
}
```

Component configuration sections

Component configuration includes several major sections. Sections in a component configuration can be in any order.

- **alarmMetrics (optional)**

A list of [metrics \(p. 345\)](#) to monitor for the component. All component types can have an alarmMetrics section.

- **logs (optional)**

A list of [logs \(p. 345\)](#) to monitor for the component. Only EC2 instances can have a logs section.

- **instances (optional)**

Nested instance configuration for the component. The following types of component can have nested instances and an instances section: ELB, ASG, and custom-grouped EC2 instances.

- **alarms (optional)**

A list of [alarms \(p. 347\)](#) to monitor for the component. All component types can have an alarm section.

The following example shows the syntax for the **instances section fragment** in JSON format.

```
{
  "alarmMetrics" : [
    list of alarm metrics
  ],
  "logs" : [
    list of logs
  ],
  "windowsEvents" : [
    list of windows events channels configurations
  ]
}
```

Metric

Defines a metric to be monitored for the component.

JSON

```
{
  "alarmMetricName" : "monitoredMetricName",
  "monitor" : true/false
}
```

Properties

- **alarmMetricName (required)**

The name of the metric to be monitored for the component. For metrics supported by Application Insights, see [Logs and metrics supported by Amazon CloudWatch Application Insights for .NET and SQL Server \(p. 355\)](#).

- **monitor (optional)**

Boolean to indicate whether to monitor the metric. The default value is `true`.

Log

Defines a log to be monitored for the component.

JSON

```
{
  "logGroupName" : "logGroupName",
  "logPath" : "logPath",
  "logType" : "logType",
  "encoding" : "encodingType",
  "monitor" : true/false
}
```

Properties

- **logGroupName (required)**

The CloudWatch log group name to be associated to the monitored log. For the log group name constraints, see [CreateLogGroup](#).

- **logPath (required for EC2 instance components; not required for components that do not use CloudWatch Agent, such as AWS Lambda)**

The path of the logs to be monitored. The log path must be an absolute Windows system file path. For more information, see [CloudWatch Agent Configuration File: Logs Section](#).

- **logType (required)**

The log type decides the log patterns against which Application Insights analyzes the log. The log type is selected from the following: SQL_SERVER/IIS/APPLICATION/DEFAULT.

- **encoding (optional)**

The type of encoding of the logs to be monitored. The specified encoding should be included in the list of [CloudWatch agent supported encodings](#). If not provided, CloudWatch Application Insights uses the default encoding type for the log type:

- For APPLICATION/DEFAULT: utf-8 encoding
- For SQL_SERVER: utf-16 encoding
- For IIS: ascii encoding

- **monitor (optional)**

Boolean that indicates whether to monitor the logs. The default value is `true`.

Windows Events

Defines Windows Events to log.

JSON

```
{
  "logGroupName" : "logGroupName",
  "eventName" : "eventName",
  "eventLevels" : ["ERROR", "WARNING", "CRITICAL", "INFORMATION", "VERBOSE"],
  "monitor" : true/false
}
```

Properties

- **logGroupName (required)**

The CloudWatch log group name to be associated to the monitored log. For the log group name constraints, see [CreateLogGroup](#).

- **eventName (required)**

The type of Windows Events to log. It is equivalent to the Windows Event log channel name. For example, System, Security, CustomEventName, etc. This field is required for each type of Windows event to log.

- **eventLevels (required)**

The levels of event to log. You must specify each level to log. Possible values include `INFORMATION`, `WARNING`, `ERROR`, `CRITICAL`, and `VERBOSE`. This field is required for each type of Windows Event to log.

- **monitor (optional)**

Boolean that indicates whether to monitor the logs. The default value is `true`.

Alarm

Defines a CloudWatch alarm to be monitored for the component.

JSON

```
{
  "alarmName" : "monitoredAlarmName",
  "severity" : HIGH/MEDIUM/LOW
}
```

Properties

- **alarmName (required)**

The name of the CloudWatch alarm to be monitored for the component.

- **severity (optional)**

Indicates the degree of outage when the alarm goes off.

Component configuration examples

The following examples show component configurations in JSON format for relevant services.

Amazon Elastic Compute Cloud (EC2) instance

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed"
    }
  ],
  "logs" : [
    {
      "logGroupName" : "my_log_group",
      "logPath" : "C:\\LogFolder\\*",
      "logType" : "APPLICATION",
      "monitor" : true
    },
  ],
  {
```

```
        "logGroupName" : "my_log_group_2",
        "logPath" : "C:\\LogFolder2\\*",
        "logType" : "IIS",
        "encoding" : "utf-8"
    }
],
"windowsEvents" : [
    {
        "logGroupName" : "my_log_group_3",
        "eventName" : "Application",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
    }, {
        "logGroupName" : "my_log_group_4",
        "eventName" : "System",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
    }
],
"alarms" : [
    {
        "alarmName" : "my_instance_alarm_1",
        "severity" : "HIGH"
    },
    {
        "alarmName" : "my_instance_alarm_2",
        "severity" : "LOW"
    }
]
}
```

Amazon Relational Database (RDS) instance

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "BurstBalance",
      "monitor" : true
    }, {
      "alarmMetricName" : "WriteThroughput",
      "monitor" : false
    }
  ],
  "alarms" : [
    {
      "alarmName" : "my_rds_instance_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Elastic Load Balancing (ELB)

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "EstimatedALBActiveConnectionCount",
    }, {
      "alarmMetricName" : "HTTPCode_Backend_5XX"
    }
  ],
  "instances" : {
    "alarmMetrics" : [
```

```
    {
      "alarmMetricName" : "CPUUtilization",
    }, {
      "alarmMetricName" : "StatusCheckFailed"
    }
  ],
  "logs" : [
    {
      "logGroupName" : "my_log_group",
      "logPath" : "C:\\LogFolder\\*",
      "logType" : "APPLICATION",
    }
  ],
  "windowsEvents" : [
    {
      "logGroupName" : "my_log_group_2",
      "eventName" : "Application",
      "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
      "monitor" : true
    }
  ]
},
"alarms" : [
  {
    "alarmName" : "my_elb_alarm",
    "severity" : "HIGH"
  }
]
}
```

Application ELB

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "ActiveConnectionCount",
    }, {
      "alarmMetricName" : "TargetResponseTime"
    }
  ],
  "instances" : {
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUUtilization",
      }, {
        "alarmMetricName" : "StatusCheckFailed"
      }
    ],
    "logs" : [
      {
        "logGroupName" : "my_log_group",
        "logPath" : "C:\\LogFolder\\*",
        "logType" : "APPLICATION",
      }
    ],
    "windowsEvents" : [
      {
        "logGroupName" : "my_log_group_2",
        "eventName" : "Application",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ]
      }
    ]
  },
}
```

```
"alarms" : [
  {
    "alarmName" : "my_alb_alarm",
    "severity" : "LOW"
  }
]
```

Amazon EC2 Auto Scaling (ASG)

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "CPUCreditBalance",
    }, {
      "alarmMetricName" : "EBSIOBalance%"
    }
  ],
  "instances" : {
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUUtilization",
      }, {
        "alarmMetricName" : "StatusCheckFailed"
      }
    ],
    "logs" : [
      {
        "logGroupName" : "my_log_group",
        "logPath" : "C:\\LogFolder\\*",
        "logType" : "APPLICATION",
      }
    ],
    "windowsEvents" : [
      {
        "logGroupName" : "my_log_group_2",
        "eventName" : "Application",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ]
      }
    ]
  },
  "alarms" : [
    {
      "alarmName" : "my_asg_alarm",
      "severity" : "LOW"
    }
  ]
}
```

Amazon Simple Queue Service (SQS)

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "ApproximateAgeOfOldestMessage"
    }, {
      "alarmMetricName" : "NumberOfEmptyReceives"
    }
  ],
  "alarms" : [
    {
      "alarmName" : "my_sqs_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

```
    }  
  ]  
}
```

Customer Grouped EC2 instances

```
{  
  "instances" : {  
    "alarmMetrics" : [  
      {  
        "alarmMetricName" : "CPUUtilization",  
      }, {  
        "alarmMetricName" : "StatusCheckFailed"  
      }  
    ],  
    "logs" : [  
      {  
        "logGroupName" : "my_log_group",  
        "logPath" : "C:\\\\LogFolder\\\\*",  
        "logType" : "APPLICATION",  
      }  
    ],  
    "windowsEvents" : [  
      {  
        "logGroupName" : "my_log_group_2",  
        "eventName" : "Application",  
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ]  
      }  
    ]  
  },  
  "alarms" : [  
    {  
      "alarmName" : "my_alarm",  
      "severity" : "MEDIUM"  
    }  
  ]  
}
```

AWS Lambda Function

```
{  
  "alarmMetrics": [  
    {  
      "alarmMetricName": "Errors",  
      "monitor": true  
    },  
    {  
      "alarmMetricName": "Throttles",  
      "monitor": true  
    },  
    {  
      "alarmMetricName": "IteratorAge",  
      "monitor": true  
    },  
    {  
      "alarmMetricName": "Duration",  
      "monitor": true  
    }  
  ],  
  "logs": [  
    {  
      "logType": "DEFAULT",  
      "monitor": true  
    }  
  ]  
}
```

```
    }  
  ]  
}
```

Amazon DynamoDB table

```
{  
  "alarmMetrics": [  
    {  
      "alarmMetricName": "SystemErrors",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "UserErrors",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "ConsumedReadCapacityUnits",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "ConsumedWriteCapacityUnits",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "ReadThrottleEvents",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "WriteThrottleEvents",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "ConditionalCheckFailedRequests",  
      "monitor": false  
    },  
    {  
      "alarmMetricName": "TransactionConflict",  
      "monitor": false  
    }  
  ],  
  "logs": []  
}
```

RDS MySQL

```
{  
  "alarmMetrics": [  
    {  
      "alarmMetricName": "CPUUtilization",  
      "monitor": true  
    }  
  ],  
  "logs": [  
    {  
      "logType": "MYSQL",  
      "monitor": true,  
    },  
    {  
      "logType": "MYSQL_SLOW_QUERY",  
      "monitor": false  
    }  
  ]  
}
```



```
}
```

Amazon S3 bucket

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "ReplicationLatency",
      "monitor" : true
    }, {
      "alarmMetricName" : "5xxErrors",
      "monitor" : true
    }, {
      "alarmMetricName" : "BytesDownloaded",
      "monitor" : true
    }
  ]
}
```

View and troubleshoot problems detected by Amazon CloudWatch Application Insights for .NET and SQL Server

An overview of problems impacting your .NET and SQL Server applications is listed under the CloudWatch Application Insights for .NET and SQL Server widget in the default overview page of the CloudWatch console. For more information, see [Getting started with Amazon CloudWatch Application Insights for .NET and SQL Server \(p. 331\)](#).

The CloudWatch Application Insights for .NET and SQL Server widget displays the following:

- The severity of the problems detected
- A summary of the problem
- The possible root cause of the problem
- The time the problem started
- The resolution status of the problem
- The affected Resource Group

To drill down into details of a specific problem, under **Problem Summary**, select the description of the problem. A detailed dashboard displays insights into the problem and related metric anomalies and snippets of log errors. From here, you can provide feedback on the relevance of the insight by selecting whether it is useful.

If a new, unconfigured resource is detected, the problem summary description takes you to the **Edit configuration** wizard to configure your new resource. If needed, you can view or edit your Resource Group configuration by choosing **View/edit configuration** in the upper right-hand corner of the detailed dashboard.

To return to the overview, choose **Back to overview**, which is next to the CloudWatch Application Insights for .NET and SQL Server detailed dashboard header.

Information Provided About Detected Problems

CloudWatch Application Insights for .NET and SQL Server provides the following information about detected problems:

- A short summary of the problem
- The start time and date of the problem
- The problem severity: High/Medium/Low
- The status of the detected problem: In-progress/Resolved
- Insights: Automatically generated insights on the detected problem and possible root cause
- Feedback on insights: Feedback you have provided about the usefulness of the insights generated by CloudWatch Application Insights for .NET and SQL Server
- Related observations: A detailed view of the metric anomalies and error snippets of relevant logs related to the problem across various application components

Feedback

You can provide feedback on the automatically generated insights on detected problems by designating them useful or not useful. Your feedback on the insights, along with your application diagnostics (metric anomalies and log exceptions), are used to improve the future detection of similar problems.

Configuration errors

CloudWatch Application Insights for .NET and SQL Server uses your configuration to create monitoring telemetries for the components. When Application Insights detects an issue with your account or your configuration, information is provided in the **Remarks** field about how to resolve the configuration issue for your application.

The following table shows suggested resolutions for specific remarks.

Remarks	Suggested Resolution	Additional notes
The quota for alarms has already been reached.	By default, each AWS account can have 5,000 CloudWatch alarms per AWS Region. See CloudWatch Limits . When throttled by this limit, CloudWatch Application Insights for .NET and SQL Server cannot create all of the required alarms to monitor your application. To resolve this, raise the account limit for CloudWatch alarms.	n/a
The quota for CloudFormation has already been reached.	Application Insights creates one CloudFormation stack for each application to manage CloudWatch agent installation and configuration for all application components. By default, each AWS account can have 200 stacks. See AWS CloudFormation Limits . To resolve this, raise the limit for CloudFormation stacks.	n/a

Remarks	Suggested Resolution	Additional notes
No SSM instance role on the following instances.	For Application Insights to be able to install and configure CloudWatch agent on application instances, AmazonSSMManagedInstanceCore and CloudWatchAgentServerPolicy policies must be attached to the instance role.	Application Insights calls the SSM DescribeInstanceInformation API to get the list of instances with SSM permission. After the role is attached to the instance, it takes time for SSM to include the instance in the DescribeInstanceInformation result. Until SSM includes the instance in the result, NO_SSM_INSTANCE_ROLE error remains present for the application.
New components may need configuration.	Application Insights detects that there are new components in the application Resource Group. To resolve this, configure the new components accordingly.	n/a

Logs and metrics supported by Amazon CloudWatch Application Insights for .NET and SQL Server

The following lists show the supported logs and metrics for CloudWatch Application Insights for .NET and SQL Server.

CloudWatch Application Insights for .NET and SQL Server supports the following logs:

- Microsoft Internet Information Services (IIS) logs
- Error log for SQL Server on EC2
- Custom .NET application logs, such as Log4Net
- Windows Event logs, including Windows logs (System, Application, and Security) and Applications and Services log
- Amazon CloudWatch Logs for AWS Lambda
- Error log and slow log for RDS MySQL and MySQL on EC2

CloudWatch Application Insights for .NET and SQL Server supports metrics for the following application components:

- [Amazon Elastic Compute Cloud \(EC2\)](#) (p. 356)
- [Elastic Load Balancer \(ELB\)](#) (p. 362)
- [Application ELB](#) (p. 362)
- [Amazon EC2 Auto Scaling Groups](#) (p. 362)
- [Amazon Simple Queue Server \(SQS\)](#) (p. 363)
- [Amazon Relational Database Service \(RDS\)](#) (p. 363)
- [AWS Lambda Function](#) (p. 364)

- [Amazon DynamoDB table](#) (p. 364)
- [Amazon S3 bucket](#) (p. 365)
- [Metrics With datapoints requirements](#) (p. 365)
- [Recommended metrics](#) (p. 370)
- [Performance Counter metrics](#) (p. 374)

Amazon Elastic Compute Cloud (EC2)

Metrics

- [CloudWatch built-in metrics](#) (p. 356)
- [CloudWatch Agent metrics \(Windows server\)](#) (p. 357)
- [CloudWatch Agent metrics \(Linux server\)](#) (p. 359)

CloudWatch built-in metrics

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPUSurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBSByteBalance%

EBSIOBalance%

EBSReadBytes

EBSReadOps

EBSWriteBytes

EBSWriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed_Instance

StatusCheckFailed_System

CloudWatch Agent metrics (Windows server)

.NET CLR Exceptions # of Exceps Thrown
.NET CLR Exceptions # of Exceps Thrown/Sec
.NET CLR Exceptions # of Filters/sec
.NET CLR Exceptions # of Finallys/sec
.NET CLR Exceptions Throw to Catch Depth/sec
.NET CLR Interop # of CCWs
.NET CLR Interop # of Stubs
.NET CLR Interop # of TLB exports/sec
.NET CLR Interop # of TLB imports/sec
.NET CLR Interop # of marshaling
.NET CLR Jit % Time in Jit
.NET CLR Jit Standard Jit Failures
.NET CLR Loading % Time Loading
.NET CLR Loading Rate of Load Failures
.NET CLR LocksAndThreads Contention Rate/sec
.NET CLR LocksAndThreads Queue Length/sec
.NET CLR Memory # Total Committed Bytes
.NET CLR Memory % Time in GC
.NET CLR Networking 4.0.0.0 HttpRequest Average Queue Time
.NET CLR Networking 4.0.0.0 HttpRequest Aborted/sec
.NET CLR Networking 4.0.0.0 HttpRequest Failed/sec
.NET CLR Networking 4.0.0.0 HttpRequest Queued/sec
ASP.NET Application Restarts
ASP.NET Applications Errors Total/Sec
ASP.NET Applications Errors Unhandled During Execution/sec
ASP.NET Applications Requests in Application Queue
ASP.NET Applications Requests/Sec
ASP.NET Request Wait Time
ASP.NET Requests Queued

HTTP Service Request Queues CurrentQueueSize

LogicalDisk % Free Space

Memory % Committed Bytes In Use

Memory Available Mbytes

Memory Pages/sec

Network Interface Bytes Total/sec

Paging File % Usage

PhysicalDisk % Disk Time

PhysicalDisk Avg. Disk Queue Length

PhysicalDisk Avg. Disk sec/Read

PhysicalDisk Avg. Disk sec/Write

PhysicalDisk Disk Read Bytes/sec

PhysicalDisk Disk Reads/sec

PhysicalDisk Disk Write Bytes/sec

PhysicalDisk Disk Writes/sec

Processor % Idle Time

Processor % Interrupt Time

Processor % Processor Time

Processor % User Time

SQLServer:Access Methods Forwarded Records/sec

SQLServer:Access Methods Full Scans/sec

SQLServer:Access Methods Page Splits/sec

SQLServer:Buffer Manager Buffer cache hit ratio

SQLServer:Buffer Manager Page life expectancy

SQLServer:General Statistics Processes blocked

SQLServer:General Statistics User Connections

SQLServer:Latches Average Latch Wait Time (ms)

SQLServer:Locks Average Wait Time (ms)

SQLServer:Locks Lock Timeouts/sec

SQLServer:Locks Lock Waits/sec

SQLServer:Locks Number of Deadlocks/sec

SQLServer:Memory Manager Memory Grants Pending

SQLServer:SQL Statistics Batch Requests/sec

SQLServer:SQL Statistics SQL Compilations/sec

SQLServer:SQL Statistics SQL Re-Compilations/sec

System Processor Queue Length

TCPv4 Connections Established

TCPv6 Connections Established

W3SVC_W3WP File Cache Flushes

W3SVC_W3WP File Cache Misses

W3SVC_W3WP Requests/Sec

W3SVC_W3WP URI Cache Flushes

W3SVC_W3WP URI Cache Misses

Web Service Bytes Received/Sec

Web Service Bytes Sent/Sec

Web Service Connection attempts/sec

Web Service Current Connections

Web Service Get Requests/sec

Web Service Post Requests/sec

CloudWatch Agent metrics (Linux server)

cpu_time_active

cpu_time_guest

cpu_time_guest_nice

cpu_time_idle

cpu_time_iowait

cpu_time_irq

cpu_time_nice

cpu_time_softirq

cpu_time_steal

cpu_time_system

cpu_time_user

cpu_usage_active

cpu_usage_guest
cpu_usage_guest_nice
cpu_usage_idle
cpu_usage_iowait
cpu_usage_irq
cpu_usage_nice
cpu_usage_softirq
cpu_usage_steal
cpu_usage_system
cpu_usage_user
disk_free
disk_inodes_free
disk_inodes_used
disk_used
disk_used_percent
diskio_io_time
diskio_iops_in_progress
diskio_read_bytes
diskio_read_time
diskio_reads
diskio_write_bytes
diskio_write_time
diskio_writes
mem_active
mem_available
mem_available_percent
mem_buffered
mem_cached
mem_free
mem_inactive
mem_used

mem_used_percent
net_bytes_rcv
net_bytes_sent
net_drop_in
net_drop_out
net_err_in
net_err_out
net_packets_rcv
net_packets_sent
netstat_tcp_close
netstat_tcp_close_wait
netstat_tcp_closing
netstat_tcp_established
netstat_tcp_fin_wait1
netstat_tcp_fin_wait2
netstat_tcp_last_ack
netstat_tcp_listen
netstat_tcp_none
netstat_tcp_syn_rcv
netstat_tcp_syn_sent
netstat_tcp_time_wait
netstat_udp_socket
processes_blocked
processes_dead
processes_idle
processes_paging
processes_running
processes_sleeping
processes_stopped
processes_total
processes_total_threads
processes_wait

processes_zombies

swap_free

swap_used

swap_used_percent

Elastic Load Balancer (ELB)

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

RequestCount

UnHealthyHostCount

Application ELB

EstimatedALBActiveConnectionCount

EstimatedALBConsumedLCUs

EstimatedALBNewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

Latency

RequestCount

SurgeQueueLength

UnHealthyHostCount

Amazon EC2 Auto Scaling Groups

CPUCreditBalance

CPUCreditUsage

CPUSurplusCreditBalance

CPU Surplus Credits Charged

CPU Utilization

Disk Read Bytes

Disk Read Ops

Disk Write Bytes

Disk Write Ops

EBS Byte Balance %

EBS IO Balance %

EBS Read Bytes

EBS Read Ops

EBS Write Bytes

EBS Write Ops

Network In

Network Out

Network Packets In

Network Packets Out

Status Check Failed

Status Check Failed_Instance

Status Check Failed_System

Amazon Simple Queue Server (SQS)

Approximate Age of Oldest Message

Approximate Number of Messages Delayed

Approximate Number of Messages Not Visible

Approximate Number of Messages Visible

Number of Empty Receives

Number of Messages Deleted

Number of Messages Received

Number of Messages Sent

Amazon Relational Database Service (RDS)

Burst Balance

CPUCreditBalance
CPUUtilization
DatabaseConnections
DiskQueueDepth
FailedSQLServerAgentJobsCount
FreeStorageSpace
FreeableMemory
NetworkReceiveThroughput
NetworkTransmitThroughput
ReadIOPS
ReadLatency
ReadThroughput
WriteIOPS
WriteLatency
WriteThroughput

AWS Lambda Function

Errors
DeadLetterErrors
Duration
Throttles
IteratorAge
ProvisionedConcurrencySpilloverInvocations

Amazon DynamoDB table

SystemErrors
UserErrors
ConsumedReadCapacityUnits
ConsumedWriteCapacityUnits
ReadThrottleEvents
WriteThrottleEvents
TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

Amazon S3 bucket

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

Metrics With datapoints requirements

For metrics without an obvious default threshold to alarm on, Application Insights waits until the metric has enough data points to predict a reasonable threshold to alarm on. The metric datapoints requirement that CloudWatch Application Insights checks before an alarm is created are:

- The metric has at least 100 datapoints from the past 15 to the past 2 days.
- The metric has at least 100 datapoints from the last day.

The following metrics follow these datapoints requirements. Note that CloudWatch agent metrics require up to one hour to create alarms.

Metrics

- [AWS/ApplicationELB \(p. 366\)](#)
- [AWS/AutoScaling \(p. 366\)](#)
- [AWS/EC2 \(p. 367\)](#)
- [AWS/ELB \(p. 367\)](#)
- [AWS/RDS \(p. 368\)](#)
- [AWS/Lambda \(p. 368\)](#)
- [AWS/SQS \(p. 368\)](#)
- [AWS/CWAgent \(p. 369\)](#)
- [AWS/DynamoDB \(p. 369\)](#)
- [AWS/S3 \(p. 370\)](#)

AWS/ApplicationELB

ActiveConnectionCount

ConsumedLCUs

HTTPCode_ELB_4XX_Count

HTTPCode_Target_2XX_Count

HTTPCode_Target_3XX_Count

HTTPCode_Target_4XX_Count

HTTPCode_Target_5XX_Count

NewConnectionCount

ProcessedBytes

TargetResponseTime

UnHealthyHostCount

AWS/AutoScaling

GroupDesiredCapacity

GroupInServiceInstances

GroupMaxSize

GroupMinSize

GroupPendingInstances

GroupStandbyInstances

GroupTerminatingInstances

GroupTotalInstances

AWS/EC2

CPUCreditBalance
CPUCreditUsage
CPUSurplusCreditBalance
CPUSurplusCreditsCharged
CPUUtilization
DiskReadBytes
DiskReadOps
DiskWriteBytes
DiskWriteOps
EBSByteBalance%
EBSIOBalance%
EBSReadBytes
EBSReadOps
EBSWriteBytes
EBSWriteOps
NetworkIn
NetworkOut
NetworkPacketsIn
NetworkPacketsOut

AWS/ELB

EstimatedALBActiveConnectionCount
EstimatedALBConsumedLCUs
EstimatedALBNewConnectionCount
EstimatedProcessedBytes
HTTPCode_Backend_4XX
HTTPCode_Backend_5XX
HealthyHostCount
Latency
RequestCount

SurgeQueueLength

UnHealthyHostCount

AWS/RDS

CPUCreditBalance

DatabaseConnections

DiskQueueDepth

FreeStorageSpace

FreeableMemory

NetworkReceiveThroughput

NetworkTransmitThroughput

ReadIOPS

ReadThroughput

WriteIOPS

WriteThroughput

AWS/Lambda

Errors

DeadLetterErrors

Duration

Throttles

IteratorAge

ProvisionedConcurrencySpilloverInvocations

AWS/SQS

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

AWS/CWAgent

LogicalDisk % Free Space

Memory % Committed Bytes In Use

Memory Available Mbytes

Network Interface Bytes Total/sec

Paging File % Usage

PhysicalDisk % Disk Time

PhysicalDisk Avg. Disk sec/Read

PhysicalDisk Avg. Disk sec/Write

PhysicalDisk Disk Read Bytes/sec

PhysicalDisk Disk Reads/sec

PhysicalDisk Disk Write Bytes/sec

PhysicalDisk Disk Writes/sec

Processor % Idle Time

Processor % Interrupt Time

Processor % Processor Time

Processor % User Time

SQLServer:Access Methods Forwarded Records/sec

SQLServer:Access Methods Page Splits/sec

SQLServer:Buffer Manager Buffer cache hit ratio

SQLServer:Buffer Manager Page life expectancy

SQLServer:General Statistics Processes blocked

SQLServer:SQL Statistics Batch Requests/sec

SQLServer:SQL Statistics SQL Compilations/sec

SQLServer:SQL Statistics SQL Re-Compilations/sec

System Processor Queue Length

TCPv4 Connections Established

TCPv6 Connections Established

AWS/DynamoDB

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents
WriteThrottleEvents
TimeToLiveDeletedItemCount
ConditionalCheckFailedRequests
TransactionConflict
ReturnedRecordsCount
PendingReplicationCount
ReplicationLatency

AWS/S3

ReplicationLatency
BytesPendingReplication
OperationsPendingReplication
4xxErrors
5xxErrors
AllRequests
GetRequests
PutRequests
DeleteRequests
HeadRequests
PostRequests
SelectRequests
ListRequests
SelectScannedBytes
SelectReturnedBytes
FirstByteLatency
TotalRequestLatency
BytesDownloaded
BytesUploaded

Recommended metrics

The following table lists the recommended metrics for each component type.

Component Type	Workload Type	Recommended Metric
EC2 instance (Windows servers)	Microsoft IIS/.NET Web Front-End	CPUUtilization StatusCheckFailed Processor % Processor Time Memory % Committed Bytes In Use Memory Available Mbytes .NET CLR Exceptions # of Exceps Thrown/Sec .NET CLR Memory # Total Committed Bytes .NET CLR Memory % Time in GC ASP.NET Applications Requests in Application Queue ASP.NET Requests Queued ASP.NET Application Restarts
	Microsoft SQL Server Database Tier	CPUUtilization StatusCheckFailed Processor % Processor Time Memory % Committed Bytes In Use Memory Available Mbytes Paging File % Usage System Processor Queue Length Network Interface Bytes Total/Sec PhysicalDisk % Disk Time SQLServer:Buffer Manager Buffer Cache Hit ratio SQLServer:Buffer Manager Page Life Expectancy SQLServer:General Statistics Processes Blocked SQLServer:General Statistics User Connections

Component Type	Workload Type	Recommended Metric
		<p>SQLServer:Locks Number of Deadlocks/Sec</p> <p>SQLServer:SQL Statistics Batch Requests/Sec</p>
	.NET workerpool/Mid-Tier	<p>CPUUtilization</p> <p>StatusCheckFailed</p> <p>Processor % Processor Time</p> <p>Memory % Committed Bytes In Use</p> <p>Memory Available Mbytes</p> <p>.NET CLR Exceptions # of Exceps Thrown/Sec</p> <p>.NET CLR Memory # Total Committed Bytes</p> <p>.NET CLR Memory % Time in GC</p>
	.NET Core Tier	<p>CPUUtilization</p> <p>StatusCheckFailed</p> <p>Processor % Processor Time</p> <p>Memory % Committed Bytes In Use</p> <p>Memory Available Mbytes</p>
EC2 instance (Linux servers)	.NET Core Tier or SQL Server Database Tier	<p>CPUUtilization</p> <p>StatusCheckFailed</p> <p>disk_used_percent</p> <p>mem_used_percent</p>
Classic ELB	Any	<p>HTTPCode_Backend_4XX</p> <p>HTTPCode_Backend_5XX</p> <p>Latency</p> <p>SurgeQueueLength</p> <p>UnHealthyHostCount</p>

Component Type	Workload Type	Recommended Metric
Application ELB	Any	HTTPCode_Target_4XX_Count HTTPCode_Target_5XX_Count TargetResponseTime UnHealthyHostCount
RDS Instance	Any	CPUUtilization ReadLatency WriteLatency BurstBalance FailedSQLServerAgentJobsCount
Lambda Function	Any	Duration Errors IteratorAge ProvisionedConcurrencySpilloverInvocations Throttles
SQS Queue	Any	ApproximateAgeOfOldestMessage ApproximateNumberOfMessagesVisible NumberOfMessagesSent
Amazon DynamoDB table	Any	SystemErrors UserErrors ConsumedReadCapacityUnits ConsumedWriteCapacityUnits ReadThrottleEvents WriteThrottleEvents ConditionalCheckFailedRequests TransactionConflict

Component Type	Workload Type	Recommended Metric
Amazon S3 bucket	Any	<p>If replication configuration with Replication Time Control (RTC) is enabled:</p> <p>ReplicationLatency</p> <p>BytesPendingReplication</p> <p>OperationsPendingReplication</p> <p>If request metrics are turned on:</p> <p>5xxErrors</p> <p>4xxErrors</p> <p>BytesDownloaded</p> <p>BytesUploaded</p>

Performance Counter metrics

Performance Counter metrics are recommended for instances only when the corresponding Performance Counter sets are installed on the Windows instances.

Performance Counter metric name	Performance Counter set name
.NET CLR Exceptions # of Exceps Thrown	.NET CLR Exceptions
.NET CLR Exceptions # of Exceps Thrown/Sec	.NET CLR Exceptions
.NET CLR Exceptions # of Filters/Sec	.NET CLR Exceptions
.NET CLR Exceptions # of Finallys/Sec	.NET CLR Exceptions
.NET CLR Exceptions Throw to Catch Depth/Sec	.NET CLR Exceptions
.NET CLR Interop # of CCWs	.NET CLR Interop
.NET CLR Interop # of Stubs	.NET CLR Interop
.NET CLR Interop # of TLB exports/Sec	.NET CLR Interop
.NET CLR Interop # of TLB imports/Sec	.NET CLR Interop
.NET CLR Interop # of Marshaling	.NET CLR Interop
.NET CLR Jit % Time in Jit	.NET CLR Jit
.NET CLR Jit Standard Jit Failures	.NET CLR Jit
.NET CLR Loading % Time Loading	.NET CLR Loading
.NET CLR Loading Rate of Load Failures	.NET CLR Loading
.NET CLR LocksAndThreads Contention Rate/Sec	.NET CLR LocksAndThreads

Performance Counter metric name	Performance Counter set name
.NET CLR LocksAndThreads Queue Length/Sec	.NET CLR LocksAndThreads
.NET CLR Memory # Total Committed Bytes	.NET CLR Memory
.NET CLR Memory % Time in GC	.NET CLR Memory
.NET CLR Networking 4.0.0.0 HttpWebRequest Average Queue Time	.NET CLR Networking 4.0.0.0
.NET CLR Networking 4.0.0.0 HttpWebRequests Aborted/Sec	.NET CLR Networking 4.0.0.0
.NET CLR Networking 4.0.0.0 HttpWebRequests Failed/Sec	.NET CLR Networking 4.0.0.0
.NET CLR Networking 4.0.0.0 HttpWebRequests Queued/Sec	.NET CLR Networking 4.0.0.0
ASP.NET Application Restarts	ASP.NET
ASP.NET Applications Errors Total/Sec	ASP.NET Applications
ASP.NET Applications Errors Unhandled During Execution/Sec	ASP.NET Applications
ASP.NET Applications Requests in Application Queue	ASP.NET Applications
ASP.NET Applications Requests/Sec	ASP.NET Applications
ASP.NET Request Wait Time	ASP.NET
ASP.NET Requests Queued	ASP.NET
HTTP Service Request Queues CurrentQueueSize	HTTP Service Request Queues
LogicalDisk % Free Space	LogicalDisk
Memory % Committed Bytes In Use	Memory
Memory Available Mbytes	Memory
Memory Pages/Sec	Memory
Network Interface Bytes Total/Sec	Network Interface
Paging File % Usage	Paging File
PhysicalDisk % Disk Time	PhysicalDisk
PhysicalDisk Avg. Disk Queue Length	PhysicalDisk
PhysicalDisk Avg. Disk Sec/Read	PhysicalDisk
PhysicalDisk Avg. Disk Sec/Write	PhysicalDisk
PhysicalDisk Disk Read Bytes/Sec	PhysicalDisk
PhysicalDisk Disk Reads/Sec	PhysicalDisk
PhysicalDisk Disk Write Bytes/Sec	PhysicalDisk

Performance Counter metric name	Performance Counter set name
PhysicalDisk Disk Writes/Sec	PhysicalDisk
Processor % Idle Time	Processor
Processor % Interrupt Time	Processor
Processor % Processor Time	Processor
Processor % User Time	Processor
SQLServer:Access Methods Forwarded Records/Sec	SQLServer:Access Methods
SQLServer:Access Methods Full Scans/Sec	SQLServer:Access Methods
SQLServer:Access Methods Page Splits/Sec	SQLServer:Access Methods
SQLServer:Buffer Manager Buffer cache hit Ratio	SQLServer:Buffer Manager
SQLServer:Buffer Manager Page life Expectancy	SQLServer:Buffer Manager
SQLServer:General Statistics Processes Blocked	SQLServer:General Statistics
SQLServer:General Statistics User Connections	SQLServer:General Statistics
SQLServer:Latches Average Latch Wait Time (ms)	SQLServer:Latches
SQLServer:Locks Average Wait Time (ms)	SQLServer:Locks
SQLServer:Locks Lock Timeouts/Sec	SQLServer:Locks
SQLServer:Locks Lock Waits/Sec	SQLServer:Locks
SQLServer:Locks Number of Deadlocks/Sec	SQLServer:Locks
SQLServer:Memory Manager Memory Grants Pending	SQLServer:Memory Manager
SQLServer:SQL Statistics Batch Requests/Sec	SQLServer:SQL Statistics
SQLServer:SQL Statistics SQL Compilations/Sec	SQLServer:SQL Statistics
SQLServer:SQL Statistics SQL Re-Compilations/Sec	SQLServer:SQL Statistics
System Processor Queue Length	System
TCPv4 Connections Established	TCPv4
TCPv6 Connections Established	TCPv6
W3SVC_W3WP File Cache Flushes	W3SVC_W3WP
W3SVC_W3WP File Cache Misses	W3SVC_W3WP
W3SVC_W3WP Requests/Sec	W3SVC_W3WP
W3SVC_W3WP URI Cache Flushes	W3SVC_W3WP
W3SVC_W3WP URI Cache Misses	W3SVC_W3WP

Performance Counter metric name	Performance Counter set name
Web Service Bytes Received/Sec	Web Service
Web Service Bytes Sent/Sec	Web Service
Web Service Connection Attempts/Sec	Web Service
Web Service Current Connections	Web Service
Web Service Get Requests/Sec	Web Service
Web Service Post Requests/Sec	Web Service

AWS Services That Publish CloudWatch Metrics

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

Service	Namespace	Documentation
Amazon API Gateway	AWS/ApiGateway	Monitor API Execution with Amazon CloudWatch
AppStream 2.0	AWS/AppStream	Monitoring Amazon AppStream 2.0 Resources
AWS AppSync	AWS/AppSync	CloudWatch Metrics
Amazon Athena	AWS/Athena	Monitoring Athena Queries with CloudWatch Metrics
AWS Billing and Cost Management	AWS/Billing	Monitoring Charges with Alerts and Notifications
ACM Private CA	AWS/ACMPrivateCA	Supported CloudWatch Metrics
AWS Chatbot	AWS/Chatbot	Monitoring AWS Chatbot with Amazon CloudWatch
Amazon CloudFront	AWS/CloudFront	Monitoring CloudFront Activity Using CloudWatch
AWS CloudHSM	AWS/CloudHSM	Getting CloudWatch Metrics
Amazon CloudSearch	AWS/CloudSearch	Monitoring an Amazon CloudSearch Domain with Amazon CloudWatch
Amazon CloudWatch Logs	AWS/Logs	Monitoring Usage with CloudWatch Metrics
AWS CodeBuild	AWS/CodeBuild	Monitoring AWS CodeBuild
Amazon Cognito	AWS/Cognito	Monitoring Amazon Cognito
Amazon Connect	AWS/Connect	Monitoring Amazon Connect in Amazon CloudWatch Metrics
AWS DataSync	AWS/DataSync	Monitoring Your Task
AWS Database Migration Service	AWS/DMS	Monitoring AWS DMS Tasks
AWS Direct Connect	AWS/DX	Monitoring with Amazon CloudWatch
Amazon DocumentDB	AWS/DocDB	Amazon DocumentDB Metrics
Amazon DynamoDB	AWS/DynamoDB	DynamoDB Metrics and Dimensions
Amazon EC2	AWS/EC2	Monitoring Your Instances Using CloudWatch

Service	Namespace	Documentation
Amazon EC2 Elastic Graphics	AWS/ ElasticGPUs	Using CloudWatch metrics to monitor Elastic Graphics
Amazon EC2 Spot Fleet	AWS/EC2Spot Fleet	CloudWatch Metrics for Spot Fleet
Amazon EC2 Auto Scaling	AWS/ AutoScaling	Monitoring Your Auto Scaling Groups and Instances Using CloudWatch
AWS Elastic Beanstalk	AWS/ ElasticBeanstalk	Publishing Amazon CloudWatch Custom Metrics for an Environment
Amazon Elastic Block Store	AWS/EBS	Amazon CloudWatch Metrics for Amazon EBS
Amazon Elastic Container Service	AWS/ECS	Amazon ECS CloudWatch Metrics
Amazon Elastic File System	AWS/EFS	Monitoring with CloudWatch
Amazon Elastic Inference	AWS/ ElasticInference	Using CloudWatch Metrics to Monitor Amazon EI
Elastic Load Balancing	AWS/ ApplicationELB	CloudWatch Metrics for Your Application Load Balancer
Elastic Load Balancing	AWS/ELB	CloudWatch Metrics for Your Classic Load Balancer
Elastic Load Balancing	AWS/NetworkELB	CloudWatch Metrics for Your Network Load Balancer
Amazon Elastic Transcoder	AWS/ ElasticTranscoder	Monitoring with Amazon CloudWatch
Amazon ElastiCache for Memcached	AWS/ ElastiCache	Monitoring Use with CloudWatch Metrics
Amazon ElastiCache for Redis	AWS/ ElastiCache	Monitoring Use with CloudWatch Metrics
Amazon Elasticsearch Service	AWS/ES	Monitoring Cluster Metrics and Statistics with CloudWatch
Amazon EMR	AWS/ ElasticMapReduce	Monitor Metrics with CloudWatch
AWS Elemental MediaConnect	AWS/ MediaConnect	Monitoring MediaConnect with Amazon CloudWatch
AWS Elemental MediaConvert	AWS/ MediaConvert	Using CloudWatch Metrics to View Metrics for AWS Elemental MediaConvert Resources

Service	Namespace	Documentation
AWS Elemental MediaPackage	AWS/MediaPackage	Monitoring AWS Elemental MediaPackage with Amazon CloudWatch Metrics
AWS Elemental MediaStore	AWS/MediaStore	Monitoring AWS Elemental MediaStore with Amazon CloudWatch Metrics
AWS Elemental MediaTailor	AWS/MediaTailor	Monitoring AWS Elemental MediaTailor with Amazon CloudWatch
Amazon EventBridge	AWS/Events	Monitoring Usage with CloudWatch Metrics
Amazon FSx for Lustre	AWS/FSx	Monitoring Amazon FSx for Lustre
Amazon FSx for Windows File Server	AWS/FSx	Monitoring Amazon FSx for Windows File Server
Amazon GameLift	AWS/GameLift	Monitor Amazon GameLift with CloudWatch
AWS Glue	AWS/Glue	Monitoring AWS Glue Using CloudWatch Metrics
AWS Ground Station	AWS/GroundStation	Metrics Using Amazon CloudWatch
Amazon Inspector	AWS/Inspector	Monitoring Amazon Inspector Using CloudWatch
AWS IoT	AWS/IoT	AWS IoT Metrics and Dimensions
AWS IoT Analytics	AWS/IoTAnalytics	Namespace, Metrics, and Dimensions
AWS IoT SiteWise	AWS/IoTSiteWise	Monitoring AWS IoT SiteWise with Amazon CloudWatch metrics
AWS IoT Things Graph	AWS/ThingsGraph	Metrics
AWS Key Management Service	AWS/KMS	Monitoring with CloudWatch
Amazon Keyspaces (for Apache Cassandra)	AWS/Cassandra	Amazon Keyspaces Metrics and Dimensions
Amazon Kinesis Data Analytics	AWS/KinesisAnalytics	Kinesis Data Analytics for SQL Applications: Monitoring with CloudWatch Kinesis Data Analytics for Apache Flink: Viewing Amazon Kinesis Data Analytics Metrics and Dimensions
Amazon Kinesis Data Firehose	AWS/Firehose	Monitoring Kinesis Data Firehose Using CloudWatch Metrics
Amazon Kinesis Data Streams	AWS/Kinesis	Monitoring Amazon Kinesis Data Streams with Amazon CloudWatch

Service	Namespace	Documentation
Amazon Kinesis Video Streams	AWS/KinesisVideo	Monitoring Kinesis Video Streams Metrics with CloudWatch
AWS Lambda	AWS/Lambda	AWS Lambda Metrics
Amazon Lex	AWS/Lex	Monitoring Amazon Lex with CloudWatch
Amazon Machine Learning	AWS/ML	Monitoring Amazon ML with CloudWatch Metrics
Amazon Managed Streaming for Apache Kafka	AWS/Kafka	Monitoring Amazon MSK with Amazon CloudWatch
Amazon MQ	AWS/AmazonMQ	Monitoring Amazon MQ Brokers Using Amazon CloudWatch
Amazon Neptune	AWS/Neptune	Monitoring Neptune with CloudWatch
AWS OpsWorks	AWS/OpsWorks	Monitoring Stacks using Amazon CloudWatch
Amazon Polly	AWS/Polly	Integrating CloudWatch with Amazon Polly
Amazon QLDB	AWS/QLDB	Amazon QLDB Metrics and Dimensions
Amazon Redshift	AWS/Redshift	Amazon Redshift Performance Data
Amazon Relational Database Service	AWS/RDS	Monitoring with Amazon CloudWatch
AWS RoboMaker	AWS/Robomaker	Monitoring AWS RoboMaker with Amazon CloudWatch
Amazon Route 53	AWS/Route53	Monitoring Amazon Route 53
Amazon SageMaker	AWS/SageMaker	Monitoring Amazon SageMaker with CloudWatch
AWS SDK Metrics for Enterprise Support	AWS/SDKMetrics	Metrics and Data Collected by AWS SDK Metrics for Enterprise Support
AWS Service Catalog	AWS/ServiceCatalog	AWS Service Catalog CloudWatch Metrics
AWS Shield Advanced	AWS/DDoSProtection	Monitoring with CloudWatch
Amazon Simple Email Service	AWS/SES	Retrieving Amazon SES Event Data from CloudWatch
Amazon Simple Notification Service	AWS/SNS	Monitoring Amazon SNS with CloudWatch
Amazon Simple Queue Service	AWS/SQS	Monitoring Amazon SQS Queues Using CloudWatch
Amazon Simple Storage Service	AWS/S3	Monitoring Metrics with Amazon CloudWatch

Service	Namespace	Documentation
Amazon Simple Workflow Service	AWS/SWF	Amazon SWF Metrics for CloudWatch
AWS Step Functions	AWS/States	Monitoring Step Functions Using CloudWatch
AWS Storage Gateway	AWS/StorageGateway	Monitoring Your Gateway and Resources
AWS Systems Manager Run Command	AWS/SSM-RunCommand	Monitoring Run Command Metrics Using CloudWatch
Amazon Textract	AWS/Textract	CloudWatch Metrics for Amazon Textract
AWS Transfer for SFTP	AWS/Transfer	AWS SFTP CloudWatch Metrics
Amazon Translate	AWS/Translate	CloudWatch Metrics and Dimensions for Amazon Translate
AWS Trusted Advisor	AWS/TrustedAdvisor	Creating Trusted Advisor Alarms Using CloudWatch
Amazon VPC	AWS/NATGateway	Monitoring Your NAT Gateway with CloudWatch
Amazon VPC	AWS/TransitGateway	CloudWatch Metrics for Your Transit Gateways
Amazon VPC	AWS/VPN	Monitoring with CloudWatch
AWS WAF	WAF	Monitoring with CloudWatch
Amazon WorkMail	AWS/WorkMail	Monitoring Amazon WorkMail with Amazon CloudWatch
Amazon WorkSpaces	AWS/WorkSpaces	Monitor Your WorkSpaces Using CloudWatch Metrics

Alarm Events and EventBridge

CloudWatch sends events to Amazon EventBridge whenever a CloudWatch alarm changes alarm state. You can use EventBridge and these events to write rules that take actions, such as notifying you, when an alarm changes state. For more information, see [What is Amazon EventBridge?](#)

Sample Events from CloudWatch

This section includes example events from CloudWatch.

State Change for a Single-Metric Alarm

```
{
  "version": "0",
  "id": "c4c1c1c9-6542-e61b-6ef0-8c4d36933a92",
  "detail-type": "CloudWatch Alarm State Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2019-10-02T17:04:40Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
  ],
  "detail": {
    "alarmName": "ServerCpuTooHigh",
    "configuration": {
      "description": "Goes into alarm when server CPU utilization is too high!",
      "metrics": [
        {
          "id": "30b6c6b2-a864-43a2-4877-c09a1afc3b87",
          "metricStat": {
            "metric": {
              "dimensions": {
                "InstanceId": "i-12345678901234567"
              },
              "name": "CPUUtilization",
              "namespace": "AWS/EC2"
            },
            "period": 300,
            "stat": "Average"
          },
          "returnData": true
        }
      ]
    },
    "previousState": {
      "reason": "Threshold Crossed: 1 out of the last 1 datapoints  
[0.0666851903306472 (01/10/19 13:46:00)] was not greater than the threshold (50.0)  
(minimum 1 datapoint for ALARM -> OK transition).",
      "reasonData": "{\n  \"version\": \"1.0\", \"queryDate\":  
\"2019-10-01T13:56:40.985+0000\", \"startDate\": \"2019-10-01T13:46:00.000+0000\", \"statistic  
\": \"Average\", \"period\": 300, \"recentDatapoints\": [0.0666851903306472], \"threshold  
\": 50.0}",
      "timestamp": "2019-10-01T13:56:40.987+0000",
      "value": "OK"
    },
    "state": {
```

```
      "reason": "Threshold Crossed: 1 out of the last 1 datapoints [99.50160229693434  
(02/10/19 16:59:00)] was greater than the threshold (50.0) (minimum 1 datapoint for OK ->  
ALARM transition).",  
      "reasonData": "{\\\"version\\\":\\\"1.0\\\",\\\"queryDate\\\":  
\\\"2019-10-02T17:04:40.985+0000\\\",\\\"startDate\\\":\\\"2019-10-02T16:59:00.000+0000\\\",\\\"statistic  
\\\":\\\"Average\\\",\\\"period\\\":300,\\\"recentDatapoints\\\":[99.50160229693434],\\\"threshold  
\\\":50.0}\"",  
      "timestamp": "2019-10-02T17:04:40.989+0000",  
      "value": "ALARM"  
    }  
  }  
}
```

State Change for a Metric Math Alarm

```
{  
  "version": "0",  
  "id": "2dde0eb1-528b-d2d5-9ca6-6d590caf2329",  
  "detail-type": "CloudWatch Alarm State Change",  
  "source": "aws.cloudwatch",  
  "account": "123456789012",  
  "time": "2019-10-02T17:20:48Z",  
  "region": "us-east-1",  
  "resources": [  
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"  
  ],  
  "detail": {  
    "alarmName": "TotalNetworkTrafficTooHigh",  
    "configuration": {  
      "description": "Goes into alarm if total network traffic exceeds 10Kb",  
      "metrics": [  
        {  
          "expression": "SUM(METRICS())",  
          "id": "e1",  
          "label": "Total Network Traffic",  
          "returnData": true  
        },  
        {  
          "id": "m1",  
          "metricStat": {  
            "metric": {  
              "dimensions": {  
                "InstanceId": "i-12345678901234567"  
              },  
              "name": "NetworkIn",  
              "namespace": "AWS/EC2"  
            },  
            "period": 300,  
            "stat": "Maximum"  
          },  
          "returnData": false  
        },  
        {  
          "id": "m2",  
          "metricStat": {  
            "metric": {  
              "dimensions": {  
                "InstanceId": "i-12345678901234567"  
              },  
              "name": "NetworkOut",  
              "namespace": "AWS/EC2"  
            },  
            "period": 300,  
            "stat": "Maximum"  
          },  
          "returnData": false  
        }  
      ]  
    }  
  }  
}
```



```

        "returnData": false
      }
    ],
    "previousState": {
      "reason": "Unchecked: Initial alarm creation",
      "timestamp": "2019-10-02T17:20:03.642+0000",
      "value": "INSUFFICIENT_DATA"
    },
    "state": {
      "reason": "Threshold Crossed: 1 out of the last 1 datapoints [45628.0 (02/10/19 17:10:00)] was greater than the threshold (10000.0) (minimum 1 datapoint for OK -> ALARM transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-02T17:20:48.551+0000\", \"startDate\": \"2019-10-02T17:10:00.000+0000\", \"period\": 300, \"recentDatapoints\": [45628.0], \"threshold\": 10000.0}\",
      "timestamp": "2019-10-02T17:20:48.554+0000",
      "value": "ALARM"
    }
  }
}

```

State Change for an Anomaly Detection Alarm

```

{
  "version": "0",
  "id": "daafc9f1-bddd-c6c9-83af-74971fcfc4ef",
  "detail-type": "CloudWatch Alarm State Change",
  "source": "aws.cloudwatch",
  "account": "123456789012",
  "time": "2019-10-03T16:00:04Z",
  "region": "us-east-1",
  "resources": ["arn:aws:cloudwatch:us-east-1:123456789012:alarm:EC2 CPU Utilization Anomaly"],
  "detail": {
    "alarmName": "EC2 CPU Utilization Anomaly",
    "state": {
      "value": "ALARM",
      "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.0 (03/10/19 15:58:00)] was less than the lower thresholds [0.020599444741798756] or greater than the upper thresholds [0.3006915352732461] (minimum 1 datapoint for OK -> ALARM transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-03T16:00:04.650+0000\", \"startDate\": \"2019-10-03T15:58:00.000+0000\", \"period\": 60, \"recentDatapoints\": [0.0], \"recentLowerThresholds\": [0.020599444741798756], \"recentUpperThresholds\": [0.3006915352732461]}\",
      "timestamp": "2019-10-03T16:00:04.653+0000"
    },
    "previousState": {
      "value": "OK",
      "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.1666666666664241 (03/10/19 15:57:00)] was not less than the lower thresholds [0.0206719426210418] or not greater than the upper thresholds [0.30076870222143803] (minimum 1 datapoint for ALARM -> OK transition).",
      "reasonData": "{\n\"version\": \"1.0\", \"queryDate\": \"2019-10-03T15:59:04.670+0000\", \"startDate\": \"2019-10-03T15:57:00.000+0000\", \"period\": 60, \"recentDatapoints\": [0.1666666666664241], \"recentLowerThresholds\": [0.0206719426210418], \"recentUpperThresholds\": [0.30076870222143803]}\",
      "timestamp": "2019-10-03T15:59:04.672+0000"
    },
    "configuration": {
      "description": "Goes into alarm if CPU Utilization is out of band",
      "metrics": [{
        "id": "m1",
        "metricStat": {
          "metric": {

```

```
        "namespace": "AWS/EC2",
        "name": "CPUUtilization",
        "dimensions": {
            "InstanceId": "i-12345678901234567"
        }
    },
    "period": 60,
    "stat": "Average"
},
"returnData": true
}, {
    "id": "ad1",
    "expression": "ANOMALY_DETECTION_BAND(m1, 0.8)",
    "label": "CPUUtilization (expected)",
    "returnData": true
}]
}
}
```

Ingesting High-Cardinality Logs and Generating Metrics with CloudWatch Embedded Metric Format

The CloudWatch embedded metric format enables you to ingest complex high-cardinality application data in the form of logs and to generate actionable metrics from them. You can embed custom metrics alongside detailed log event data, and CloudWatch automatically extracts the custom metrics so that you can visualize and alarm on them, for real-time incident detection. Additionally, the detailed log events associated with the extracted metrics can be queried using CloudWatch Logs Insights to provide deep insights into the root causes of operational events.

Embedded metric format helps you to generate actionable custom metrics from ephemeral resources such as Lambda functions and containers. By using the embedded metric format to send logs from these ephemeral resources, you can now easily create custom metrics without having to instrument or maintain separate code, while gaining powerful analytical capabilities on your log data.

When using the embedded metric format, you can generate your logs using a client library— for more information, see [Using the Client Libraries to Generate Embedded Metric Format Logs \(p. 388\)](#). Alternatively, you can manually construct the logs and submit them using the PutLogEvents API or the CloudWatch agent.

Charges are incurred for logs ingestion and archival, and custom metrics that are generated. For more information, see [Amazon CloudWatch Pricing](#).

Note

Be careful when configuring your metric extraction as it impacts your custom metric usage and corresponding bill. If you unintentionally create metrics based on high-cardinality dimensions (such as `requestId`), the embedded metric format will by design create a custom metric corresponding to each unique dimension combination. For more information, see [Dimensions](#).

Topics

- [Generating Logs Using the Embedded Metric Format \(p. 387\)](#)
- [Viewing Your Metrics and Logs in the Console \(p. 400\)](#)

Generating Logs Using the Embedded Metric Format

You can generate embedded metric format logs with the following methods:

- Generate and send the logs by using the open-sourced client libraries
- Manually generate the logs, and then use the CloudWatch agent or the PutLogEvents API to send the logs

If you use one of the manual methods, the logs must follow the defined JSON format.

Topics

- [Using the Client Libraries to Generate Embedded Metric Format Logs \(p. 388\)](#)
- [Manually Generating Embedded Metric Format Logs \(p. 388\)](#)

Using the Client Libraries to Generate Embedded Metric Format Logs

Amazon provides open-sourced client libraries which you can use to create embedded metric format logs. Currently those libraries are available in Node.js and Python. Support for other languages is planned.

The libraries and the instructions for how to use them are located on Github. Use the links in the following list.

- [Node.js](#)
- [Python](#)

Manually Generating Embedded Metric Format Logs

To send embedded metric format logs that you have manually created, you can use the PutLogEvents API or the CloudWatch agent.

If you use either of these methods, you must follow the embedded metric format specification.

Topics

- [Specification: Embedded Metric Format \(p. 388\)](#)
- [Using the PutLogEvents API to Send Manually-Created Embedded Metric Format Logs \(p. 393\)](#)
- [Using the CloudWatch Agent to Send Embedded Metric Format Logs \(p. 395\)](#)

Specification: Embedded Metric Format

The CloudWatch embedded metric format is a JSON specification used to instruct CloudWatch Logs to automatically extract metric values embedded in structured log events. You can use CloudWatch to graph and create alarms on the extracted metric values.

Embedded Metric Format Specification Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this format specification are to be interpreted as described in [Key Words RFC2119](#).

The terms "JSON", "JSON text", "JSON value", "member", "element", "object", "array", "number", "string", "boolean", "true", "false", and "null" in this format specification are to be interpreted as defined in [JavaScript Object Notation RFC8259](#).

Embedded Metric Format Specification PutLogEvents Request Format

Clients **MUST** use the following log format header when sending an embedded metric format document using the CloudWatch Logs [PutLogEvents API](#):

```
x-amzn-logs-format: json/emf
```

On Lambda, you do not need to set this header yourself. Writing JSON to standard out in the embedded metric format is sufficient. While Lambda may prepend log events with metadata such as timestamp and request id, a valid embedded metric format document after this metadata is considered valid.

Embedded Metric Format Document Structure

This section describes the structure of an embedded metric format document, which is identified by the log-format header `x-amzn-logs-format: json/emf`. Embedded metric format documents are defined in [JavaScript Object Notation RFC8259](#).

Unless otherwise noted, objects defined by this specification **MUST NOT** contain any additional members. Members not recognized by this specification **MUST** be ignored. Members defined in this specification are case-sensitive.

The embedded metric format is subject to the same limits as standard CloudWatch Logs events and are limited to a maximum size of 256 KB.

Root Node

The LogEvent message **MUST** be a valid JSON object with no additional data at the beginning or end of the LogEvent message string. For more information about the LogEvent structure, see [InputLogEvent](#).

Embedded metric format documents **MUST** contain the following top-level member on the root node. This is a [Metadata Object \(p. 389\)](#) object.

```
{
  "_aws": {
    "CloudWatchMetrics": [ ... ]
  }
}
```

The root node **MUST** contain all [??? \(p. 391\)](#) members defined by the references in the [MetricDirective Object \(p. 390\)](#).

The root node **MAY** contain any other members that are not included in the above requirements. The values of these members **MUST** be valid JSON types.

Metadata Object

The `_aws` member can be used to represent metadata about the payload that informs downstream services how they should process the LogEvent. The value **MUST** be an object and **MUST** contain the following members:

- **CloudWatchMetrics**— An array of [MetricDirective Object \(p. 390\)](#) used to instruct CloudWatch to extract metrics from the root node of the LogEvent.

```
{
  "_aws": {
    "CloudWatchMetrics": [ ... ]
  }
}
```

- **Timestamp**— A number representing the time stamp used for metrics extracted from the event. Values **MUST** be expressed as the number of milliseconds after Jan 1, 1970 00:00:00 UTC.

```
{
  "_aws": {
    "Timestamp": 1559748430481
  }
}
```

MetricDirective Object

The MetricDirective object instructs downstream services that the LogEvent contains metrics that will be extracted and published to CloudWatch. MetricDirectives MUST contain the following members:

- **Namespace**— A string representing the CloudWatch namespace for the metric.
- **Dimensions**— A [DimensionSet Array \(p. 390\)](#).
- **Metrics**— An array of [MetricDefinition \(p. 390\)](#) objects. This array MUST NOT contain more than 100 MetricDefinition objects.

DimensionSet Array

A DimensionSet is an array of strings containing the dimension keys that will be applied to all metrics in the document. The values within this array MUST also be members on the root-node—referred to as the [Target Members \(p. 391\)](#)

A DimensionSet MUST NOT contain more than 9 dimension keys.

The target member MUST have a string value. The target member defines a dimension that will be published as part of the metric identity. Every DimensionSet used creates a new metric in CloudWatch. For more information about dimensions, see [Dimension](#) and [Dimensions](#).

```
{
  "_aws": {
    "CloudWatchMetrics": [
      {
        "Dimensions": [ [ "functionVersion" ] ],
        ...
      }
    ]
  },
  "functionVersion": "$LATEST"
}
```

Note

Be careful when configuring your metric extraction as it impacts your custom metric usage and corresponding bill. If you unintentionally create metrics based on high-cardinality dimensions (such as `requestId`), the embedded metric format will by design create a custom metric corresponding to each unique dimension combination. For more information, see [Dimensions](#).

MetricDefinition Object

A MetricDefinition is an object that MUST contain the following member:

- **Name**— A string [Reference Values \(p. 391\)](#) to a metric [Target Members \(p. 391\)](#). Metric targets MUST be either a numeric value or an array of numeric values.

A MetricDefinition object MAY contain the following member:

- **Unit**— An OPTIONAL string value representing the unit of measure for the corresponding metric. Values SHOULD be valid CloudWatch metric units. For information about valid units, see [MetricDatum](#). If a value is not provided, then a default value of NONE is assumed.

```
{
  "_aws": {
    "CloudWatchMetrics": [
      {

```

```
    "Metrics": [
      {
        "Name": "Time",
        "Unit": "Milliseconds"
      }
    ],
    ...
  }
]
},
"Time": 1
}
```

Reference Values

Reference values are string values that reference [Target Members \(p. 391\)](#) members on the root node. These references should NOT be confused with the JSON Pointers described in [RFC6901](#). Target values cannot be nested.

Target Members

Valid targets MUST be members on the root node and cannot be nested objects. For example, a `_reference_` value of `"A.a"` MUST match the following member:

```
{ "A.a" }
```

It MUST NOT match the nested member:

```
{ "A": { "a" } }
```

Valid values of target members depend on what is referencing them. For example, a metric target MUST be a numeric value or an array of numeric values while dimension targets MUST be string values.

Embedded Metric Format Example and JSON Schema

The following is a valid example of embedded metric format.

```
{
  "_aws": {
    "Timestamp": 1574109732004,
    "CloudWatchMetrics": [
      {
        "Namespace": "lambda-function-metrics",
        "Dimensions": [ [ "functionVersion" ] ],
        "Metrics": [
          {
            "Name": "time",
            "Unit": "Milliseconds"
          }
        ]
      }
    ]
  },
  "functionVersion": "$LATEST",
  "time": 100,
  "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```

You can use the following schema to validate embedded metric format documents.

```
{
```

```
"type": "object",
"title": "Root Node",
"required": [
  "_aws"
],
"properties": {
  "_aws": {
    "$id": "#/properties/_aws",
    "type": "object",
    "title": "Metadata",
    "required": [
      "Timestamp",
      "CloudWatchMetrics"
    ],
    "properties": {
      "Timestamp": {
        "$id": "#/properties/_aws/properties/Timestamp",
        "type": "integer",
        "title": "The Timestamp Schema",
        "examples": [
          1565375354953
        ]
      },
      "CloudWatchMetrics": {
        "$id": "#/properties/_aws/properties/CloudWatchMetrics",
        "type": "array",
        "title": "MetricDirectives",
        "items": {
          "$id": "#/properties/_aws/properties/CloudWatchMetrics/items",
          "type": "object",
          "title": "MetricDirective",
          "required": [
            "Namespace",
            "Dimensions",
            "Metrics"
          ],
          "properties": {
            "Namespace": {
              "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Namespace",
              "type": "string",
              "title": "CloudWatch Metrics Namespace",
              "examples": [
                "MyApp"
              ],
              "pattern": "^(.*)$",
              "minLength": 1
            },
            "Dimensions": {
              "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions",
              "type": "array",
              "title": "The Dimensions Schema",
              "minItems": 1,
              "items": {
                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions/items",
                "type": "array",
                "title": "DimensionSet",
                "minItems": 1,
                "maxItems": 9,
                "items": {
                  "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items/items",
                  "type": "string",
                  "title": "DimensionReference",
```



```

        "examples": [
            {
                "Operation":
                ],
                "pattern": "^(.*)$",
                "minItems": 1
            }
        }
    },
    "Metrics": {
        "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics",
        "type": "array",
        "title": "MetricDefinitions",
        "items": {
            "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics/items",
            "type": "object",
            "title": "MetricDefinition",
            "required": [
                "Name"
            ],
            "properties": {
                "Name": {
                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Name",
                    "type": "string",
                    "title": "MetricName",
                    "examples": [
                        "ProcessingLatency"
                    ],
                    "pattern": "^(.*)$"
                },
                "Unit": {
                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Unit",
                    "type": "string",
                    "title": "MetricUnit",
                    "examples": [
                        "Milliseconds"
                    ],
                    "pattern": "^(Seconds|Microseconds|
Milliseconds|Bytes|Kilobytes|Megabytes|Gigabytes|Terabytes|Bits|Kilobits|Megabits|Gigabits|
Terabits|Percent|Count|Bytes\\|Second|Kilobytes\\|Second|Megabytes\\|Second|Gigabytes\\|
Second|Terabytes\\|Second|Bits\\|Second|Kilobits\\|Second|Megabits\\|Second|Gigabits\\|
Second|Terabits\\|Second|Count\\|Second|None)$"
                }
            }
        }
    }
}

```

Using the PutLogEvents API to Send Manually-Created Embedded Metric Format Logs

You can send embedded metric format logs to CloudWatch Logs using the CloudWatch Logs PutLogEvents API. When calling PutLogEvents, you need to include the following HTTP header to instruct CloudWatch Logs that the metrics should be extracted.

```
x-amzn-logs-format: json/emf
```

The following is a full example using the AWS SDK for Java 2.x:

```
package org.example.basicapp;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.InputLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.PutLogEventsRequest;

import java.util.Collections;

public class EmbeddedMetricsExample {
    public static void main(String[] args) {

        final String usage = "To run this example, supply a region id (eg. us-east-1), log group, and stream name as command line arguments"
            + "Ex: PutLogEvents <region-id> <log-group-name> <stream-name>";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String regionId = args[0];
        String logGroupName = args[1];
        String logStreamName = args[2];

        CloudWatchLogsClient logsClient =
            CloudWatchLogsClient.builder().region(Region.of(regionId)).build();

        // A sequence token is required to put a log event in an existing stream.
        // Look up the stream to find its sequence token.
        String sequenceToken = getNextSequenceToken(logsClient, logGroupName,
            logStreamName);

        // Build a JSON log using the EmbeddedMetricFormat.
        long timestamp = System.currentTimeMillis();
        String message = "{" +
            "  \"_aws\": {" +
            "    \"Timestamp\": " + timestamp + "," +
            "    \"CloudWatchMetrics\": [" +
            "      {" +
            "        \"Namespace\": \"MyApp\", " +
            "        \"Dimensions\": [\"Operation\"], [\"Operation\",
\\\"Cell\\\"], " +
            "        \"Metrics\": [{ \"Name\": \"ProcessingLatency\",
\\\"Unit\": \"Milliseconds\" }]" +
            "      }" +
            "    ]" +
            "  }, " +
            "  \"Operation\": \"Aggregator\", " +
            "  \"Cell\": \"001\", " +
            "  \"ProcessingLatency\": 100" +
            "}";

        InputLogEvent inputLogEvent = InputLogEvent.builder()
            .message(message)
            .timestamp(timestamp)
            .build();
```

```
// Specify the request parameters.
PutLogEventsRequest putLogEventsRequest = PutLogEventsRequest.builder()
    .overrideConfiguration(builder ->
        // provide the log-format header of json/emf
        builder.headers(Collections.singletonMap("x-amzn-logs-
format", Collections.singletonList("json/emf"))))
    .logEvents(Collections.singletonList(inputLogEvent))
    .logGroupName(logGroupName)
    .logStreamName(logStreamName)
    .sequenceToken(sequenceToken)
    .build();

logsClient.putLogEvents(putLogEventsRequest);

System.out.println("Successfully put CloudWatch log event");
}

private static String getNextSequenceToken(CloudWatchLogsClient logsClient, String
logGroupName, String logStreamName) {
    DescribeLogStreamsRequest logStreamRequest =
DescribeLogStreamsRequest.builder()
    .logGroupName(logGroupName)
    .logStreamNamePrefix(logStreamName)
    .build();

    DescribeLogStreamsResponse describeLogStreamsResponse =
logsClient.describeLogStreams(logStreamRequest);

    // Assume that a single stream is returned since a specific stream name was
    // specified in the previous request.
    return
describeLogStreamsResponse.logStreams().get(0).uploadSequenceToken();
}
}
```

Using the CloudWatch Agent to Send Embedded Metric Format Logs

To use this method, first install the CloudWatch agent for the services you want to send embedded metric format logs from, and then you can begin sending the events.

The CloudWatch agent must be version 1.230621.0 or later.

Note

You do not need to install the CloudWatch agent to send logs from Lambda functions. Lambda function timeouts are not handled automatically. This means that if your function times out before the metrics get flushed, then the metrics for that invocation will not be captured.

Installing the CloudWatch Agent

Install the CloudWatch agent for each service which is to send embedded metric format logs.

Installing the CloudWatch Agent on EC2

First, install the CloudWatch agent on the instance. For more information, see [Installing the CloudWatch Agent \(p. 237\)](#).

Once you have installed the agent, configure the agent to listen on a UDP or TCP port for the embedded metric format logs. The following is an example of this configuration that listens on the default socket `tcp:25888`. For more information about agent configuration, see [Manually Create or Edit the CloudWatch Agent Configuration File \(p. 275\)](#).

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

Installing the CloudWatch Agent on Amazon ECS

The easiest way to deploy the CloudWatch agent on Amazon ECS is to run it as a sidecar, defining it in the same task definition as your application.

Create Agent Configuration File

Create your CloudWatch agent configuration file locally. In this example, the relative file path will be `amazon-cloudwatch-agent.json`.

For more information about agent configuration, see [Manually Create or Edit the CloudWatch Agent Configuration File \(p. 275\)](#).

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

Push Configuration to SSM Parameter Store

Enter the following command to push the CloudWatch agent configuration file to the AWS Systems Manager (SSM) Parameter Store.

```
aws ssm put-parameter \
  --name "cwagentconfig" \
  --type "String" \
  --value "`cat amazon-cloudwatch-agent.json`" \
  --region "${region}"
```

Configure The Task Definition

Configure your task definition to use the CloudWatch Agent and expose the TCP or UDP port. The sample task definition that you should use depends on your networking mode.

Notice that the `webapp` specifies the `AWS_EMF_AGENT_ENDPOINT` environment variable. This is used by the library and should point to the endpoint that the agent is listening on. Additionally, the `cwagent` specifies the `CW_CONFIG_CONTENT` as a `"valueFrom"` parameter that points to the SSM configuration that you created in the previous step.

The following is an example for bridge mode. When bridge mode networking is enabled, the agent needs to be linked to your application using the `links` parameter and addressed using the container name.

```
{
  "containerDefinitions": [
    {
      "name": "webapp",
      "links": [ "cwagent" ],
      "image": "my-org/web-app:latest",
      "essential": true,
```

```
        "memory": 256,
        "cpu": 256,
        "environment": [{
            "name": "AWS_EMF_AGENT_ENDPOINT",
            "value": "tcp://cwagent:25888"
        }],
    },
    {
        "name": "cwagent",
        "mountPoints": [],
        "image": "amazon/cloudwatch-agent:latest",
        "essential": true,
        "memory": 256,
        "cpu": 256,
        "portMappings": [{
            "protocol": "tcp",
            "containerPort": 25888
        }],
        "environment": [{
            "name": "CW_CONFIG_CONTENT",
            "value": "cwagentconfig"
        }],
    }
],
}
```

The following is an example for host mode or awsvpc mode. When running on these network modes, the agent can be addressed over localhost.

```
{
  "containerDefinitions": [
    {
      "name": "webapp",
      "image": "my-org/web-app:latest",
      "essential": true,
      "memory": 256,
      "cpu": 256,
      "environment": [{
        "name": "AWS_EMF_AGENT_ENDPOINT",
        "value": "tcp://127.0.0.1:25888"
      }],
    },
    {
      "name": "cwagent",
      "mountPoints": [],
      "image": "amazon/cloudwatch-agent:latest",
      "essential": true,
      "memory": 256,
      "cpu": 256,
      "portMappings": [{
        "protocol": "tcp",
        "containerPort": 25888
      }],
      "environment": [{
        "name": "CW_CONFIG_CONTENT",
        "value": "cwagentconfig"
      }],
    }
  ],
}
```

Note

In awsvpc mode, you must either give a public IP address to the VPC (Fargate only), set up a NAT gateway, or set up a CloudWatch Logs VPC endpoint. For more information about setting

up a NAT, see [NAT Gateways](#). For more information about setting up a CloudWatch Logs VPC endpoint, see [Using CloudWatch Logs with Interface VPC Endpoints](#). The following is an example of how to assign a public IP address to a task that uses the Fargate launch type.

```
aws ecs run-task \
--cluster {{cluster-name}} \
--task-definition cwagent-fargate \
--region {{region}} \
--launch-type FARGATE \
--network-configuration
  "awsvpcConfiguration={subnets=[{{subnetId}}],securityGroups=[{{sgId}}],assignPublicIp=ENABLED}"
```

Ensure Permissions

Ensure the IAM role executing your tasks has permission to read from the SSM Parameter Store. You can add this permission by attaching the **AmazonSSMReadOnlyAccess** policy. To do so, enter the following command.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess \
--role-name CWAgentECSExecutionRole
```

Installing the CloudWatch Agent on Amazon EKS

Parts of this process can be skipped if you have already installed CloudWatch Container Insights on this cluster.

Permissions

If you have not already installed Container Insights, then first ensure that your Amazon EKS nodes have the appropriate IAM permissions. They should have the **CloudWatchAgentServerPolicy** attached. For more information, see [Verify Prerequisites](#) (p. 165).

Create ConfigMap

Create a ConfigMap for the agent. The ConfigMap also tells the agent to listen on a TCP or UDP port. Use the following ConfigMap.

```
# cwagent-emf-configmap.yaml
apiVersion: v1
data:
  # Any changes here must not break the JSON format
  cwagentconfig.json: |
    {
      "agent": {
        "omit_hostname": true
      },
      "logs": {
        "metrics_collected": {
          "emf": { }
        }
      }
    }
kind: ConfigMap
metadata:
  name: cwagentemfconfig
  namespace: default
```

If you have already installed Container Insights, add the following "emf": { } line to your existing ConfigMap.

Apply the ConfigMap

Enter the following command to apply the ConfigMap.

```
kubectl apply -f cwagent-emf-configmap.yaml
```

Deploy the Agent

To deploy the CloudWatch agent as a sidecar, add the agent to your pod definition, as in the following example.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  namespace: default
spec:
  containers:
    # Your container definitions go here
    - name: web-app
      image: my-org/web-app:latest
    # CloudWatch Agent configuration
    - name: cloudwatch-agent
      image: amazon/cloudwatch-agent:latest
      imagePullPolicy: Always
      resources:
        limits:
          cpu: 200m
          memory: 100Mi
        requests:
          cpu: 200m
          memory: 100Mi
      volumeMounts:
        - name: cwagentconfig
          mountPath: /etc/cwagentconfig
      ports:
    # this should match the port configured in the ConfigMap
    - protocol: TCP
      hostPort: 25888
      containerPort: 25888
  volumes:
    - name: cwagentconfig
      configMap:
        name: cwagentemfconfig
```

Using the CloudWatch Agent to Send Embedded Metric Format Logs

Once you have the CloudWatch agent installed and running you can send the embedded metric format logs over TCP or UDP. There are two requirements when sending the logs over the agent:

- The logs must contain a `LogGroupName` key that tells the agent which log group to use.
- Each log event must be on a single line. In other words, a log event cannot contain the newline (`\n`) character.

The log events must also follow the embedded metric format specification. For more information, see [Specification: Embedded Metric Format \(p. 388\)](#).

The following is an example of sending log events manually from a Linux bash shell. You can instead use the UDP socket interfaces provided by your programming language of choice.

```
echo '{"_aws":{"Timestamp":1574109732004,"LogGroupName":"Foo","CloudWatchMetrics":
[{"Namespace":"MyApp","Dimensions":[["Operation"]],"Metrics":
[{"Name":"ProcessingLatency","Unit":"Milliseconds"}]}]}',"Operation":"Aggregator","ProcessingLatency":10
\
> /dev/udp/0.0.0.0/25888
```

Viewing Your Metrics and Logs in the Console

After you generate embedded metric format logs that extract metrics, you can use the CloudWatch console to view the metrics. Embedded metrics have the dimensions that you specified when you generated the logs. Also, embedded metrics that you generated using the client libraries have the following default dimensions:

- ServiceType
- ServiceName
- LogGroup

To view metrics that were generated from embedded metric format logs

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Select a namespace that you specified for your embedded metrics when you generated them. If you used the client libraries to generate the metrics and did not specify a namespace, then select **aws-embedded-metrics**. This is the default namespace for embedded metrics generated using the client libraries.
4. Select a metric dimension (for example, **ServiceName**).
5. The **All metrics** tab displays all metrics for that dimension in the namespace. You can do the following:
 - a. To sort the table, use the column heading.
 - b. To graph a metric, select the check box next to the metric. To select all metrics, select the check box in the heading row of the table.
 - c. To filter by resource, choose the resource ID and then choose **Add to search**.
 - d. To filter by metric, choose the metric name and then choose **Add to search**.

Querying Logs Using CloudWatch Logs Insights

You can query the detailed log events associated with the extracted metrics by using CloudWatch Logs Insights to provide deep insights into the root causes of operational events. One of the benefits of extracting metrics from your logs is that you can filter your logs later by the unique metric (metric name plus unique dimension set) and metric values, to get context on the events that contributed to the aggregated metric value

For example, to get an impacted request id or x-ray trace id, you could run the following query in CloudWatch Logs Insights.

```
filter Latency > 1000 and Operation = "Aggregator"
| fields RequestId, TraceId
```

You can also perform query-time aggregation on high-cardinality keys, such as finding the customers impacted by an event. The following example illustrates this.


```
filter Latency > 1000 and Operation = "Aggregator"  
| stats count() by CustomerId
```

For more information, see [Analyzing Log Data with CloudWatch Logs Insights](#)

Monitor Applications Using AWS SDK Metrics

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

Important

SDK Metrics is available only to customers with an Enterprise Support subscription. For more information, see [Amazon CloudWatch Support Center](#).

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

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

Topics

- [Metrics and Data Collected by AWS SDK Metrics for Enterprise Support \(p. 403\)](#)
- [Set Up SDK Metrics \(p. 405\)](#)

Supported Versions

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

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

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

Supported SDK	SDK Metrics documentation
AWS CLI 1.16.84 or later	SDK Metrics The instructions for the AWS CLI installation are the same as for SDK for Python (Boto 3).
AWS SDK for C++ 1.7.47 or later	SDK Metrics
AWS SDK for Go 1.25.18 or later	SDK Metrics in the AWS SDK for Go
AWS SDK for Java 1.11.523 or later (AWS SDK for Java 2.x is not yet supported)	Enabling AWS SDK Metrics for Enterprise Support
AWS SDK for JavaScript in Node.js 2.387 or later	SDK Metrics in the AWS SDK for JavaScript

Supported SDK	SDK Metrics documentation
AWS SDK for .NET 3.3.440 or later	Enabling SDK Metrics
AWS SDK for PHP 3.85.0 or later	SDK Metrics in the AWS SDK for PHP Version 3
AWS SDK for Python (Boto 3) 1.9.78 or later	SDK Metrics
AWS SDK for Ruby 3.45.0 or later	SDK Metrics in the AWS SDK for Ruby

Metrics and Data Collected by AWS SDK Metrics for Enterprise Support

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

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

Data	Type
Network Error Messages	String/map
Source IP Address	String
Destination IP Address	String

The following table lists the metrics generated by SDK Metrics, published in the `AWS/SDKMetrics` namespace. These metrics are available only to customers who have the Enterprise support plan. For these customers, the metrics can be accessed through AWS support personnel and technical account managers.

The metrics are published at one-minute intervals.

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

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

You can use the following dimensions with SDK Metrics.

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

Set Up SDK Metrics

To set up SDK Metrics, you perform some steps with your SDK and some with CloudWatch agent.

To set up SDK Metrics

1. Create an application with an AWS SDK that supports SDK Metrics.
2. Host your project on an Amazon EC2 instance or in your local environment.
3. Install and use the latest version of your SDK.
4. Install the CloudWatch agent on the EC2 instance or the local environment that is running an application that you want to receive metrics for. For more information, see [Installing the CloudWatch Agent \(p. 237\)](#).
5. Configure the CloudWatch agent for SDK Metrics. For more information, see [Configure the CloudWatch Agent for SDK Metrics \(p. 406\)](#).
6. Authorize SDK Metrics to collect and send metrics. For more information, see [Set IAM Permissions for SDK Metrics \(p. 407\)](#).
7. Enable SDK Metrics. There are multiple methods to do this, depending on which SDK you're using. One method that is universal for all SDKs is to add the following to your environment variables:

```
export AWS_CSM_ENABLED=true
```

For information about the other methods, see your SDK documentation as shown in the following table.

Supported SDK	SDK Metrics documentation
AWS CLI 1.16.84 or later	SDK Metrics The instructions for the AWS CLI installation are the same as for SDK for Python (Boto 3).
AWS SDK for C++ 1.7.47 or later	SDK Metrics

Supported SDK	SDK Metrics documentation
AWS SDK for Go 1.16.18 or later	SDK Metrics in the AWS SDK for Go
AWS SDK for Java 1.11.523 or later (AWS SDK for Java 2.x is not yet supported)	Enabling AWS SDK Metrics for Enterprise Support
AWS SDK for JavaScript in Node.js 2.387 or later	SDK Metrics in the AWS SDK for JavaScript
AWS SDK for .NET 3.3.440 or later	Enabling SDK Metrics
AWS SDK for PHP 3.85.0 or later	SDK Metrics in the AWS SDK for PHP Version 3
AWS SDK for Python (Boto 3) 1.9.78 or later	SDK Metrics
AWS SDK for Ruby 3.45.0 or later	SDK Metrics in the AWS SDK for Ruby

Configure the CloudWatch Agent for SDK Metrics

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

Topics

- [Configure the CloudWatch Agent for SDK Metrics Using AWS Systems Manager \(p. 406\)](#)
- [Configure the CloudWatch Agent for SDK Metrics Manually \(p. 407\)](#)

Configure the CloudWatch Agent for SDK Metrics Using AWS Systems Manager

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

To configure SDK Metrics using SSM

1. Open the Systems Manager console at <https://console.aws.amazon.com/systems-manager/>.
2. In the navigation pane, choose **Run Command**.
3. In the **Command document** list, choose the button next to **AWS-UpdateSSMAgent**.
4. In the **Targets** area, choose the instance where you installed the CloudWatch agent.
5. In the navigation pane, choose **Parameter Store**.
6. Choose **Create Parameter**.
7. Do the following:
 - a. Name the parameter `AmazonCSM`.
 - b. Select type `string`.
 - c. For **Value**, enter `{"csm": {"memory_limit_in_mb": 20, "port": 31000}}`.
8. Select **Create Parameter**.

To complete the configuration, see your SDK documentation.

Configure the CloudWatch Agent for SDK Metrics Manually

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

Create Your CloudWatch Agent Configuration File

If you don't already have a CloudWatch agent configuration file, create one. For more information, see [Create the CloudWatch Agent Configuration File \(p. 270\)](#).

Add the SDK Metrics Configuration

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

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

For example:

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

You can now start the CloudWatch agent using this configuration file. For more information, see [Start the CloudWatch Agent \(p. 256\)](#).

To complete the configuration, see your SDK documentation.

Set IAM Permissions for SDK Metrics

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

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy, JSON**.
4. Enter the following inline policy into the content window. Ignore the warning about the non-supported action.

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

```
        "Resource": "*"
    }
  ]
}
```

5. Name the policy **AmazonSDKMetrics**.
6. Attach the policy to the IAM user or role that is attached to the EC2 instance that is running the CloudWatch agent.

Service Quotas Integration and Usage Metrics

You can use CloudWatch to proactively manage your AWS service quotas. CloudWatch usage metrics provide visibility into your account's usage of resources and API operations.

You can use these metrics to visualize your current service usage on CloudWatch graphs and dashboards. You can use a CloudWatch metric math function to display the service quotas for those resources on your graphs. You can also configure alarms that alert you when your usage approaches a service quota.

For more information about service quotas, see [What Is Service Quotas](#) in the *Service Quotas User Guide*.

Currently, the following services provide usage metrics that you can use to evaluate your service quotas:

- AWS CloudHSM
- [Amazon CloudWatch](#)
- Amazon DynamoDB
- [Amazon EC2](#)
- [Amazon Elastic Container Registry](#)
- AWS Key Management Service
- [Amazon Kinesis Data Firehose](#)
- [AWS Robomaker](#)

All of these services publish their usage metrics in the `AWS/Usage` metric namespace.

Topics

- [Visualizing Your Service Quotas and Setting Alarms](#) (p. 409)
- [CloudWatch Usage Metrics](#) (p. 410)

Visualizing Your Service Quotas and Setting Alarms

You can use the CloudWatch console to visualize your service quotas, and see how your current usage compares. You can also set alarms to be notified when you approach a quota.

To visualize a service quota and optionally set an alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics** tab, choose **Usage**, then choose **By AWS Resource**.

The list of service quota usage metrics appears.

4. Select the check box next to one of the metrics.

The graph displays your current usage of that AWS resource.

5. To add your service quota to the graph, do the following:

- a. Choose the **Graphed metrics** tab.
- b. Choose **Math expression, Start with an empty expression**. Then in the new row, under **Details**, enter **SERVICE_QUOTA(m1)**.

A new line is added to the graph, displaying the service quota for the resource represented in the metric.
6. To see your current usage as a percentage of the quota, add a new expression or change the current **SERVICE_QUOTA** expression. For the new expression, use **m1/SERVICE_QUOTA(m1)*100**
7. (Optional) To set an alarm that notifies you if you approach the service quota, do the following:
 - a. On the **m1/SERVICE_QUOTA(m1)*100** row, under **Actions**, choose the alarm icon. It looks like a bell.

The alarm creation page appears.
 - b. Under **Conditions**, ensure that **Threshold type** is **Static** and **Whenever Expression1 is** is set to **Greater**. Under **than**, enter **80**. This creates an alarm that goes into ALARM state when your usage exceeds 80 percent of the quota.
 - c. Choose **Next**.
 - d. On the next page, select an Amazon SNS topic or create a new one. This topic is notified when the alarm goes to ALARM state. Then choose **Next**.
 - e. On the next page, enter a name and description for the alarm, and then choose **Next**.
 - f. Choose **Create alarm**.

CloudWatch Usage Metrics

CloudWatch collects metrics that track the usage of some AWS resources. These metrics correspond to AWS service quotas. Tracking these metrics can help you proactively manage your quotas. For more information, see [Service Quotas Integration and Usage Metrics \(p. 409\)](#).

Service quota usage metrics are in the `AWS/Usage` namespace and are collected every minute.

Currently, the only metric name in this namespace that CloudWatch publishes is `CallCount`. This metric is published with the dimensions `Resource`, `Service`, and `Type`. The `Resource` dimension specifies the name of the API operation being tracked. For example, the `CallCount` metric with the dimensions `"Service": "CloudWatch"`, `"Type": "API"` and `"Resource": "PutMetricData"` indicates the number of times the CloudWatch `PutMetricData` API operation has been called in your account.

The `CallCount` metric does not have a specified unit. The most useful statistic for the metric is `SUM`, which represents the total operation count for the 1-minute period.

Metrics

Metric	Description
<code>CallCount</code>	The number of specified operations performed in your account.

Dimensions

Dimension	Description
<code>Service</code>	The name of the AWS service containing the resource. For CloudWatch usage metrics, the value for this dimension is <code>CloudWatch</code> .

Dimension	Description
Class	The class of resource being tracked. CloudWatch API usage metrics use this dimension with a value of <code>None</code> .
Type	The type of resource being tracked. Currently, when the <code>Service</code> dimension is <code>CloudWatch</code> , the only valid value for <code>Type</code> is <code>API</code> .
Resource	The name of the API operation. Valid values include the following: <code>DeleteAlarms</code> , <code>DeleteDashboards</code> , <code>DescribeAlarmHistory</code> , <code>DescribeAlarms</code> , <code>GetDashboard</code> , <code>GetMetricData</code> , <code>GetMetricStatistics</code> , <code>ListMetrics</code> , <code>PutDashboard</code> , and <code>PutMetricData</code>

CloudWatch Tutorials

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

Scenarios

- [Monitor Your Estimated Charges \(p. 412\)](#)
- [Publish Metrics \(p. 414\)](#)

Scenario: Monitor Your Estimated Charges Using CloudWatch

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

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

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

Tasks

- [Step 1: Enable Billing Alerts \(p. 412\)](#)
- [Step 2: Create a Billing Alarm \(p. 413\)](#)
- [Step 3: Check the Alarm Status \(p. 413\)](#)
- [Step 4: Edit a Billing Alarm \(p. 414\)](#)
- [Step 5: Delete a Billing Alarm \(p. 414\)](#)

Step 1: Enable Billing Alerts

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

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

Requirements

- You must be signed in as either an AWS account root user or an IAM user whose access to the Billing and Cost Management console has been activated.

- For consolidated billing accounts, billing data for each linked account can be found by logging in as the paying account. You can view billing data for total estimated charges and estimated charges by service for each linked account as well as for the consolidated account.

To enable monitoring of your estimated charges

1. Open the Billing and Cost Management console at <https://console.aws.amazon.com/billing/>.
2. In the navigation pane, choose **Preferences**.
3. Select **Receive Billing Alerts**.
4. Choose **Save preferences**.

Step 2: Create a Billing Alarm

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

Note

This procedure uses the simple options. To use the advanced options, see [Creating a Billing Alarm \(p. 101\)](#) in *Create a Billing Alarm to Monitor Your Estimated AWS Charges*.

To create a billing alarm

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

Tip

The graph shows a current estimate of your charges that you can use to set an appropriate amount.

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

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

8. Choose **Create Alarm**.

Step 3: Check the Alarm Status

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

To check the alarm status

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the Region to US East (N. Virginia). Billing metric data is stored in this Region and reflects worldwide charges.

3. In the navigation pane, choose **Alarms**.
4. Select the check box next to the alarm. Until the subscription is confirmed, it is shown as "Pending confirmation". After the subscription is confirmed, refresh the console to show the updated status.

Step 4: Edit a Billing Alarm

For example, you may want to increase the amount of money you spend with AWS each month from \$200 to \$400. You can edit your existing billing alarm and increase the monetary amount that must be exceeded before the alarm is triggered.

To edit a billing alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the Region to US East (N. Virginia). Billing metric data is stored in this Region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**.
4. Select the check box next to the alarm and choose **Actions, Modify**.
5. For **Whenever my total AWS charges for the month exceed**, specify the new amount that must be exceeded to trigger the alarm and send an email notification.
6. Choose **Save Changes**.

Step 5: Delete a Billing Alarm

If you no longer need your billing alarm, you can delete it.

To delete a billing alarm

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. If necessary, change the Region to US East (N. Virginia). Billing metric data is stored in this Region and reflects worldwide charges.
3. In the navigation pane, choose **Alarms**.
4. Select the check box next to the alarm and choose **Actions, Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Scenario: Publish Metrics to CloudWatch

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

Tasks

- [Step 1: Define the Data Configuration \(p. 415\)](#)
- [Step 2: Add Metrics to CloudWatch \(p. 415\)](#)
- [Step 3: Get Statistics from CloudWatch \(p. 416\)](#)
- [Step 4: View Graphs with the Console \(p. 416\)](#)

Step 1: Define the Data Configuration

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

You publish several data points that collectively represent three hours of latency data. The raw data comprises 15 request latency readings distributed over three hours. Each reading is in milliseconds:

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

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

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

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

Step 2: Add Metrics to CloudWatch

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

To publish data points to CloudWatch

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

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

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

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

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

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

Step 3: Get Statistics from CloudWatch

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

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

The following is example output:

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

Step 4: View Graphs with the Console

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

To view graphs of your statistics on the console

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

For more information, see [Graphing Metrics](#) (p. 42).

Tagging Your Amazon CloudWatch Resources

A *tag* is a custom attribute label that you or AWS assigns to an AWS resource. Each tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, or `Project`). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, `111122223333` or `Production`). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

Tags help you do the following:

- Identify and organize your AWS resources. Many AWS services support tagging, so you can assign the same tag to resources from different services to indicate that the resources are related. For example, you can assign the same tag to a CloudWatch rule that you assign to an EC2 instance.
- Track your AWS costs. You activate these tags on the AWS Billing and Cost Management dashboard. AWS uses the tags to categorize your costs and deliver a monthly cost allocation report to you. For more information, see [Use Cost Allocation Tags](#) in the [AWS Billing and Cost Management User Guide](#).

The following sections provide more information about tags for CloudWatch.

Supported Resources in CloudWatch

The following resources in CloudWatch support tagging:

- Alarms – You can tag alarms using the [tag-resource](#) AWS CLI command and the [TagResource](#) API.
- Canaries – You can tag canaries using the CloudWatch console. For more information, see [Creating a Canary](#) (p. 107).
- Contributor Insights rules – You can tag Contributor Insights rules when you create them by using the [put-insight-rule](#) AWS CLI command and the [PutInsightRule](#) API. You can add tags to existing rules by using the [tag-resource](#) AWS CLI command and the [TagResource](#) API.

For information about adding and managing tags, see [Managing Tags](#) (p. 417).

Managing Tags

Tags consist of the `Key` and `Value` properties on a resource. You can use the CloudWatch console, the AWS CLI, or the CloudWatch API to add, edit, or delete the values for these properties. For information about working with tags, see the following:

- [TagResource](#), [UntagResource](#), and [ListTagsForResource](#) in the *Amazon CloudWatch API Reference*
- [tag-resource](#), [untag-resource](#), and [list-tags-for-resource](#) in the *Amazon CloudWatch CLI Reference*
- [Working with Tag Editor](#) in the *Resource Groups User Guide*

Tag Naming and Usage Conventions

The following basic naming and usage conventions apply to using tags with CloudWatch resources:

- Each resource can have a maximum of 50 tags.
- For each resource, each tag key must be unique, and each tag key can have only one value.
- The maximum tag key length is 128 Unicode characters in UTF-8.
- The maximum tag value length is 256 Unicode characters in UTF-8.
- Allowed characters are letters, numbers, spaces representable in UTF-8, and the following characters: `. : * = @ _ / -` (hyphen).
- Tag keys and values are case sensitive. As a best practice, decide on a strategy for capitalizing tags and consistently implement that strategy across all resource types. For example, decide whether to use `Costcenter`, `costcenter`, or `CostCenter` and use the same convention for all tags. Avoid using similar tags with inconsistent case treatment.
- The `aws:` prefix is prohibited for tags because it's reserved for AWS use. You can't edit or delete tag keys or values with this prefix. Tags with this prefix don't count against your tags per resource limit.

Security in Amazon CloudWatch

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to Amazon WorkSpaces, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using Amazon CloudWatch. It shows you how to configure Amazon CloudWatch to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your CloudWatch resources.

Contents

- [Data Protection in Amazon CloudWatch \(p. 419\)](#)
- [Identity and Access Management for Amazon CloudWatch \(p. 420\)](#)
- [Compliance Validation for Amazon CloudWatch \(p. 446\)](#)
- [Resilience in Amazon CloudWatch \(p. 446\)](#)
- [Infrastructure Security in Amazon CloudWatch \(p. 446\)](#)
- [Using CloudWatch and CloudWatch Synthetics with Interface VPC Endpoints \(p. 447\)](#)
- [Security Considerations for Synthetics Canaries \(p. 450\)](#)

Data Protection in Amazon CloudWatch

Amazon CloudWatch conforms to the AWS [shared responsibility model](#), which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.

- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields or metadata, such as function names and tags. Any data that you enter into metadata might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

Encryption in Transit

CloudWatch uses end-to-end encryption of data in transit.

Identity and Access Management for Amazon CloudWatch

Access to Amazon CloudWatch requires credentials. Those credentials must have permissions to access AWS resources, such as retrieving CloudWatch metric data about your cloud resources. The following sections provide details about how you can use [AWS Identity and Access Management \(IAM\)](#) and CloudWatch to help secure your resources by controlling who can access them:

- [Authentication](#) (p. 420)
- [Access Control](#) (p. 421)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your *AWS account user credentials* and they provide complete access to all of your AWS resources.

Important

For security reasons, we recommend that you use the AWS account user credentials only to create an *administrator*, which is an *IAM user* with full permissions to your AWS account. Then, you can use this administrator to create other IAM users and roles with limited permissions. For more information, see [IAM Best Practices](#) and [Creating an Admin User and Group](#) in the *IAM User Guide*.

- **IAM user** – An [IAM user](#) is an identity within your AWS account that has specific custom permissions (for example, permissions to view metrics in CloudWatch). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and AWS CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. CloudWatch supports *Signature Version 4*, a protocol for authenticating inbound API requests. For

more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

- **IAM role** – An [IAM role](#) is another IAM identity you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
 - **Federated user access** – Instead of creating an IAM user, you can use preexisting identities from AWS Directory Service, your enterprise user directory, or a web identity provider (IdP). These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an IdP. For more information, see [Federated Users and Roles](#) in the *IAM User Guide*.
 - **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles](#) in the *IAM User Guide*.
 - **AWS service access** – You can use an IAM role in your account to grant an AWS service the permissions needed to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
 - **Applications running on Amazon EC2** – Instead of storing access keys within the EC2 instance for use by applications running on the instance and making API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see [Using Roles for Applications on Amazon EC2](#) in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access CloudWatch resources. For example, you must have permissions to create CloudWatch dashboard widgets, view metrics, and so on.

The following sections describe how to manage permissions for CloudWatch. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your CloudWatch Resources](#) (p. 422)
- [Using Identity-Based Policies \(IAM Policies\) for CloudWatch](#) (p. 425)
- [Amazon CloudWatch Permissions Reference](#) (p. 439)

CloudWatch Dashboard Permissions Update

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

- `cloudwatch:GetDashboard`
- `cloudwatch:ListDashboards`
- `cloudwatch:PutDashboard`
- `cloudwatch>DeleteDashboards`

To access CloudWatch dashboards, you need one of the following:

- The **AdministratorAccess** policy.
- The **CloudWatchFullAccess** policy.
- A custom policy that includes one or more of these specific permissions:
 - `cloudwatch:GetDashboard` and `cloudwatch:ListDashboards` to be able to view dashboards
 - `cloudwatch:PutDashboard` to be able to create or modify dashboards
 - `cloudwatch>DeleteDashboards` to be able to delete dashboards

For more information for changing permissions for an IAM user using policies, see [Changing Permissions for an IAM User](#).

For more information about CloudWatch permissions, see [Amazon CloudWatch Permissions Reference](#) (p. 439).

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

Overview of Managing Access Permissions to Your CloudWatch Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note

An *account administrator* (or administrator IAM user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- [CloudWatch Resources and Operations](#) (p. 422)
- [Understanding Resource Ownership](#) (p. 423)
- [Managing Access to Resources](#) (p. 423)
- [Specifying Policy Elements: Actions, Effects, and Principals](#) (p. 424)
- [Specifying Conditions in a Policy](#) (p. 425)

CloudWatch Resources and Operations

You can restrict access to specific alarms and dashboards by using their Amazon Resource Names (ARNs) in your policies. For more information, see [Actions, Resources, and Condition Keys for Amazon CloudWatch](#) in the *IAM User Guide*.

You use an * (asterisk) as the resource when writing a policy to control access to CloudWatch actions. For example:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["cloudwatch:GetMetricStatistics", "cloudwatch:ListMetrics"],
    "Resource": "*",
    "Condition": {
      "Bool": {
        "aws:SecureTransport": "true"
      }
    }
  }]
}
```

For more information about ARNs, see [ARNs in IAM User Guide](#). For information about CloudWatch Logs ARNs, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#) in the *Amazon Web Services General Reference*. For an example of a policy that covers CloudWatch actions, see [Using Identity-Based Policies \(IAM Policies\) for CloudWatch](#) (p. 425).

Action	ARN (with region)	ARN (for use with IAM role)
Stop	arn:aws:automate:us-east-1:ec2:stop	arn:aws:swf:us-east-1: <i>customer-account</i> :action/actions/AWS_EC2.InstanceId.Stop/1.0
Terminate	arn:aws:automate:us-east-1:ec2:terminate	arn:aws:swf:us-east-1: <i>customer-account</i> :action/actions/AWS_EC2.InstanceId.Terminate/1.0
Reboot	n/a	arn:aws:swf:us-east-1: <i>customer-account</i> :action/actions/AWS_EC2.InstanceId.Reboot/1.0
Recover	arn:aws:automate:us-east-1:ec2:recover	n/a

Understanding Resource Ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the [principal entity](#) (that is, the AWS account root user, an IAM user, or an IAM role) that authenticates the resource creation request. CloudWatch does not have any resources that you can own.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of CloudWatch. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in

the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as identity-based policies (IAM policies) and policies attached to a resource are referred to as resource-based policies. CloudWatch supports only identity-based policies.

Topics

- [Identity-Based Policies \(IAM Policies\)](#) (p. 424)
- [Resource-Based Policies \(IAM Policies\)](#) (p. 424)

Identity-Based Policies (IAM Policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an Amazon CloudWatch resource, such as metrics, you can attach a permissions policy to a user or group that the user belongs to.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in account A can create a role to grant cross-account permissions to another AWS account (for example, account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in account A.
 2. Account A administrator attaches a trust policy to the role identifying account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in account B. Doing this allows users in account B to create or access resources in account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

For more information about using identity-based policies with CloudWatch, see [Using Identity-Based Policies \(IAM Policies\) for CloudWatch](#) (p. 425). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

Resource-Based Policies (IAM Policies)

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an Amazon S3 bucket to manage access permissions to that bucket. CloudWatch doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, and Principals

For each CloudWatch resource, the service defines a set of API operations. To grant permissions for these API operations, CloudWatch defines a set of actions that you can specify in a policy. Some API operations can require permissions for more than one action in order to perform the API operation. For more information about resources and API operations, see [CloudWatch Resources and Operations](#) (p. 422) and CloudWatch [Actions](#).

The following are the basic policy elements:

- **Resource** – Use an Amazon Resource Name (ARN) to identify the resource that the policy applies to. CloudWatch does not have any resources for you to control using policies resources, so use

the wildcard character (*) in IAM policies. For more information, see [CloudWatch Resources and Operations \(p. 422\)](#).

- **Action** – Use action keywords to identify resource operations that you want to allow or deny. For example, the `cloudwatch:ListMetrics` permission allows the user permissions to perform the `ListMetrics` operation.
- **Effect** – You specify the effect, either allow or deny, when the user requests the specific action. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). CloudWatch doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see [AWS IAM JSON Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the CloudWatch API actions and the resources that they apply to, see [Amazon CloudWatch Permissions Reference \(p. 439\)](#).

Specifying Conditions in a Policy

When you grant permissions, you can use the access policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

To express conditions, you use predefined condition keys. For a list of context keys supported by each AWS service and a list of AWS-wide policy keys, see [AWS Service Actions and Condition Context Keys](#) and [Global and IAM Condition Context Keys](#) in the *IAM User Guide*.

Using Identity-Based Policies (IAM Policies) for CloudWatch

This topic provides examples of identity-based policies that demonstrate how an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and thereby grant permissions to perform operations on CloudWatch resources.

Important

We recommend that you first review the introductory topics that explain the basic concepts and options available to manage access to your CloudWatch resources. For more information, see [Access Control \(p. 421\)](#).

The sections in this topic cover the following:

- [Permissions Required to Use the CloudWatch Console \(p. 426\)](#)
- [AWS Managed \(Predefined\) Policies for CloudWatch \(p. 428\)](#)
- [Customer Managed Policy Examples \(p. 429\)](#)

The following shows an example of a permissions policy.

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

```
{
  "Effect": "Allow",
  "Action": ["cloudwatch:GetMetricStatistics", "cloudwatch:ListMetrics"],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "true"
    }
  }
}
```

This sample policy has one statement that grants permissions to a group for two CloudWatch actions (`cloudwatch:GetMetricStatistics` and `cloudwatch:ListMetrics`), but only if the group uses SSL with the request (`"aws:SecureTransport": "true"`). For more information about the elements within an IAM policy statement, see [Specifying Policy Elements: Actions, Effects, and Principals](#) (p. 424) and [IAM Policy Elements Reference](#) in *IAM User Guide*.

Permissions Required to Use the CloudWatch Console

For a user to work with the CloudWatch console, that user must have a minimum set of permissions that allow the user to describe other AWS resources in their AWS account. The CloudWatch console requires permissions from the following services:

- Amazon EC2 Auto Scaling
- CloudTrail
- CloudWatch
- CloudWatch Events
- CloudWatch Logs
- Amazon EC2
- Amazon ES
- IAM
- Kinesis
- Lambda
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon SWF
- X-Ray, if you are using the ServiceLens feature

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy. To ensure that those users can still use the CloudWatch console, also attach the `CloudWatchReadOnlyAccess` managed policy to the user, as described in [AWS Managed \(Predefined\) Policies for CloudWatch](#) (p. 428).

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the CloudWatch API.

The full set of permissions required to work with the CloudWatch console are listed below:

- `application-autoscaling:DescribeScalingPolicies`
- `autoscaling:DescribeAutoScalingGroups`
- `autoscaling:DescribePolicies`
- `cloudtrail:DescribeTrails`

- cloudwatch:DeleteAlarms
- cloudwatch:DescribeAlarmHistory
- cloudwatch:DescribeAlarms
- cloudwatch:GetMetricData
- cloudwatch:GetMetricStatistics
- cloudwatch:ListMetrics
- cloudwatch:PutMetricAlarm
- cloudwatch:PutMetricData
- ec2:DescribeInstances
- ec2:DescribeTags
- ec2:DescribeVolumes
- es:DescribeElasticsearchDomain
- es:ListDomainNames
- events:DeleteRule
- events:DescribeRule
- events:DisableRule
- events:EnableRule
- events:ListRules
- events:PutRule
- iam:AttachRolePolicy
- iam:CreateRole
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:ListAttachedRolePolicies
- iam:ListRoles
- kinesis:DescribeStream
- kinesis:ListStreams
- lambda:AddPermission
- lambda:CreateFunction
- lambda:GetFunctionConfiguration
- lambda:ListAliases
- lambda:ListFunctions
- lambda:ListVersionsByFunction
- lambda:RemovePermission
- logs:CancelExportTask
- logs:CreateExportTask
- logs:CreateLogGroup
- logs:CreateLogStream
- logs>DeleteLogGroup
- logs>DeleteLogStream
- logs>DeleteMetricFilter
- logs>DeleteRetentionPolicy
- logs>DeleteSubscriptionFilter
- logs:DescribeExportTasks
- logs:DescribeLogGroups

- logs:DescribeLogStreams
- logs:DescribeMetricFilters
- logs:DescribeQueries
- logs:DescribeSubscriptionFilters
- logs:FilterLogEvents
- logs:GetLogGroupFields
- logs:GetLogRecord
- logs:GetLogEvents
- logs:GetQueryResults
- logs:PutMetricFilter
- logs:PutRetentionPolicy
- logs:PutSubscriptionFilter
- logs:StartQuery
- logs:StopQuery
- logs:TestMetricFilter
- s3:CreateBucket
- s3:ListBucket
- sns:CreateTopic
- sns:GetTopicAttributes
- sns:ListSubscriptions
- sns:ListTopics
- sns:SetTopicAttributes
- sns:Subscribe
- sns:Unsubscribe
- sqs:GetQueueAttributes
- sqs:GetQueueUrl
- sqs:ListQueues
- sqs:SetQueueAttributes
- swf:CreateAction
- swf:DescribeAction
- swf:ListActionTemplates
- swf:RegisterAction
- swf:RegisterDomain
- swf:UpdateAction

Additionally, to view the service map in ServiceLens, you need `AWSXrayReadOnlyAccess`

AWS Managed (Predefined) Policies for CloudWatch

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so that you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to CloudWatch:

- **CloudWatchFullAccess** – Grants full access to CloudWatch.

- **CloudWatchReadOnlyAccess** – Grants read-only access to CloudWatch.
- **CloudWatchActionsEC2Access** – Grants read-only access to CloudWatch alarms and metrics in addition to Amazon EC2 metadata. Grants access to the Stop, Terminate, and Reboot API actions for EC2 instances.

Note

You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

You can also create your own custom IAM policies to allow permissions for CloudWatch actions and resources. You can attach these custom policies to the IAM users or groups that require those permissions.

AWS Managed (Predefined) Policies for CloudWatch Synthetics

The following AWS managed policies, which you can attach to users in your account, are specific to CloudWatch Synthetics:

- **CloudWatchSyntheticsFullAccess** or **CloudWatchSyntheticsReadOnlyAccess** – To view canary details and the results of canary test runs.
- **AmazonS3ReadOnlyAccess** and **CloudWatchReadOnlyAccess** – To be able to read all Synthetics data in the CloudWatch console.
- **AWSLambdaReadOnlyAccess** – To be able to view the source code used by canaries.
- **CloudWatchSyntheticsFullAccess** – Enables you to create canaries. Additionally, to create a canary that will have a new IAM role created for it, you also need the following inline policy statement:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
        "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
      ]
    }
  ]
}
```

Important

Granting a user the `iam:CreateRole`, `iam:CreatePolicy`, and `iam:AttachRolePolicy` permissions gives that user full administrative access to your AWS account. For example, a user with these permissions can create a policy that has full permissions for all resources, and attach that policy to any role. Be very careful about who you grant these permissions to.

For information about attaching policies and granting permissions to users, see [Changing Permissions for an IAM User](#) and [To embed an inline policy for a user or role](#).

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various CloudWatch actions. These policies work when you are using the CloudWatch API, AWS SDKs, or the AWS CLI.

Examples

- [Example 1: Allow User Full Access to CloudWatch \(p. 430\)](#)
- [Example 2: Allow Read-Only Access to CloudWatch \(p. 430\)](#)
- [Example 3: Stop or Terminate an Amazon EC2 Instance \(p. 430\)](#)

Example 1: Allow User Full Access to CloudWatch

The following policy allows a user to access all CloudWatch actions, CloudWatch Logs actions, Amazon SNS actions, and read-only access to Amazon EC2 Auto Scaling.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:*",
        "logs:*",
        "sns:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example 2: Allow Read-Only Access to CloudWatch

The following policy allows a user read-only access to CloudWatch and view Amazon EC2 Auto Scaling actions, CloudWatch metrics, CloudWatch Logs data, and alarm-related Amazon SNS data.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "logs:Get*",
        "logs:Describe*",
        "sns:Get*",
        "sns:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example 3: Stop or Terminate an Amazon EC2 Instance

The following policy allows an CloudWatch alarm action to stop or terminate an EC2 instance. In the sample below, the GetMetricStatistics, ListMetrics, and DescribeAlarms actions are optional. It is recommended that you include these actions to ensure that you have correctly stopped or terminated the instance.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Action": [
      "cloudwatch:PutMetricAlarm",
      "cloudwatch:GetMetricStatistics",
      "cloudwatch:ListMetrics",
      "cloudwatch:DescribeAlarms"
    ],
    "Sid": "0000000000000000",
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "ec2:DescribeInstanceStatus",
      "ec2:DescribeInstances",
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Sid": "0000000000000000",
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

Using Condition Keys to Limit Access to CloudWatch Namespaces

Use IAM condition keys to limit users to publishing metrics only in the CloudWatch namespaces that you specify.

Allowing Publishing in One Namespace Only

The following policy limits the user to publishing metrics only in the namespace named `MyCustomNamespace`.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Resource": "*",
    "Action": "cloudwatch:PutMetricData",
    "Condition": {
      "StringEquals": {
        "cloudwatch:namespace": "MyCustomNamespace"
      }
    }
  }
}
```

Excluding Publishing from a Namespace

The following policy allows the user to publish metrics in any namespace except for `CustomNamespace2`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "*",
      "Action": "cloudwatch:PutMetricData"
    },
    {
      "Effect": "Deny",
      "Resource": "*",
      "Action": "cloudwatch:PutMetricData",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": "CustomNamespace2"
        }
      }
    }
  ]
}
```

Using Service-Linked Roles for CloudWatch

Amazon CloudWatch uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to CloudWatch. Service-linked roles are predefined by CloudWatch and include all the permissions that the service requires to call other AWS services on your behalf.

One service-linked role in CloudWatch makes setting up CloudWatch alarms that can terminate, stop, or reboot an Amazon EC2 instance without requiring you to manually add the necessary permissions. Another service-linked role enables a monitoring account to access CloudWatch data from other accounts that you specify, to build cross-account cross-Region dashboards.

CloudWatch defines the permissions of these service-linked roles, and unless defined otherwise, only CloudWatch can assume the role. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete the roles only after first deleting their related resources. This restriction protects your CloudWatch resources because you can't inadvertently remove permissions to access the resources.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-Linked Role Permissions for CloudWatch Alarms

CloudWatch uses the service-linked role named **AWSServiceRoleForCloudWatchEvents** – CloudWatch uses this service-linked role to perform Amazon EC2 alarm actions.

The **AWSServiceRoleForCloudWatchEvents** service-linked role trusts the CloudWatch Events service to assume the role. CloudWatch Events invokes the terminate, stop, or reboot instance actions when called upon by the alarm.

The **AWSServiceRoleForCloudWatchEvents** service-linked role permissions policy allows CloudWatch Events to complete the following actions on Amazon EC2 instances:

- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:RecoverInstances`

- `ec2:DescribeInstanceRecoveryAttribute`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceState`

The **AWSServiceRoleForCloudWatchCrossAccount** service-linked role permissions policy allows CloudWatch to complete the following actions:

- `sts:AssumeRole`

Service-Linked Role Permissions for CloudWatch Cross-Account Cross-Region

CloudWatch uses the service-linked role named **AWSServiceRoleForCloudWatchCrossAccount** – CloudWatch uses this role to access CloudWatch data in other AWS accounts that you specify. The SLR only provides the assume role permission to allow the CloudWatch service to assume the role in the sharing account. It is the sharing role that provides access to data.

The **AWSServiceRoleForCloudWatchCrossAccount** service-linked role trusts the CloudWatch service to assume the role.

Creating a Service-Linked Role for CloudWatch

You do not need to manually create either of these service-linked roles. The first time you create an alarm in the AWS Management Console, the IAM CLI, or the IAM API, CloudWatch creates **AWSServiceRoleForCloudWatchEvents** for you. When you first enable an account to be a monitoring account for cross-account cross-Region functionality, CloudWatch creates **AWSServiceRoleForCloudWatchCrossAccount** for you.

For more information, see [Creating a Service-Linked Role](#) in the *IAM User Guide*.

Editing a Service-Linked Role for CloudWatch

CloudWatch does not allow you to edit the **AWSServiceRoleForCloudWatchEvents** or **AWSServiceRoleForCloudWatchCrossAccount** roles. After you create these roles, you cannot change their names because various entities might reference these roles. However, you can edit the description of these roles using IAM.

Editing a Service-Linked Role Description (IAM Console)

You can use the IAM console to edit the description of a service-linked role.

To edit the description of a service-linked role (console)

1. In the navigation pane of the IAM console, choose **Roles**.
2. Choose the name of the role to modify.
3. To the far right of **Role description**, choose **Edit**.
4. Type a new description in the box, and choose **Save**.

Editing a Service-Linked Role Description (AWS CLI)

You can use IAM commands from the AWS Command Line Interface to edit the description of a service-linked role.

To change the description of a service-linked role (AWS CLI)

1. (Optional) To view the current description for a role, use the following commands:

```
$ aws iam get-role --role-name role-name
```

Use the role name, not the ARN, to refer to roles with the AWS CLI commands. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, you refer to the role as **myrole**.

2. To update a service-linked role's description, use the following command:

```
$ aws iam update-role-description --role-name role-name --description description
```

Editing a Service-Linked Role Description (IAM API)

You can use the IAM API to edit the description of a service-linked role.

To change the description of a service-linked role (API)

1. (Optional) To view the current description for a role, use the following command:

`GetRole`

2. To update a role's description, use the following command:

`UpdateRoleDescription`

Deleting a Service-Linked Role for CloudWatch

If you no longer have alarms that automatically stop, terminate, or reboot EC2 instances, we recommend that you delete the `AWSServiceRoleForCloudWatchEvents` role.

That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session in the IAM console

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**. Choose the name (not the check box) of the `AWSServiceRoleForCloudWatchEvents` role.
3. On the **Summary** page for the selected role, choose **Access Advisor** and review the recent activity for the service-linked role.

Note

If you are unsure whether CloudWatch is using the `AWSServiceRoleForCloudWatchEvents` role, try to delete the role. If the service is using the role, then the deletion fails and you can view the Regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

Deleting a Service-Linked Role (IAM Console)

You can use the IAM console to delete a service-linked role.

To delete a service-linked role (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**. Select the check box next to the name of the role you want to delete, not the name or row itself.
3. For **Role actions**, choose **Delete role**.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. To proceed, choose **Yes, Delete**.
5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, the deletion task can succeed or fail after you submit the role for deletion. If the task fails, choose **View details** or **View Resources** from the notifications to learn why the deletion failed. If the deletion fails because there are resources in the service that are being used by the role, then the reason for the failure includes a list of resources.

Deleting a Service-Linked Role (AWS CLI)

You can use IAM commands from the AWS Command Line Interface to delete a service-linked role.

To delete a service-linked role (AWS CLI)

1. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the `deletion-task-id` from the response to check the status of the deletion task. Type the following command to submit a service-linked role deletion request:

```
$ aws iam delete-service-linked-role --role-name AWSServiceRoleForCloudWatchEvents
```

2. Type the following command to check the status of the deletion task:

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

The status of the deletion task can be `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Deleting a Service-Linked Role (IAM API)

You can use the IAM API to delete a service-linked role.

To delete a service-linked role (API)

1. To submit a deletion request for a service-linked roll, call `DeleteServiceLinkedRole`. In the request, specify the role name that you want to delete.

Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the `DeletionTaskId` from the response to check the status of the deletion task.

2. To check the status of the deletion, call `GetServiceLinkedRoleDeletionStatus`. In the request, specify the `DeletionTaskId`.

The status of the deletion task can be `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Using Service-Linked Roles for CloudWatch Application Insights for .NET and SQL Server

CloudWatch Application Insights uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to CloudWatch Application Insights for .NET and SQL Server. Service-linked roles are predefined by CloudWatch Application Insights for .NET and SQL Server and include all of the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up CloudWatch Application Insights for .NET and SQL Server easier because you don't have to manually add the necessary permissions. CloudWatch Application Insights for .NET and SQL Server defines the permissions of its service-linked roles, and unless defined otherwise, only CloudWatch Application Insights for .NET and SQL Server can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** link to view the service-linked role documentation for that service.

Service-Linked Role Permissions for CloudWatch Application Insights for .NET and SQL Server

CloudWatch Application Insights for .NET and SQL Server uses the service-linked role named **ApplicationInsights-role**. Application Insights uses this role to perform operations such as analyzing the Resource Groups of the customer, creating CloudFormation stacks to create alarms on metrics, and configuring the CloudWatch Agent on EC2 instances.

The role permissions policy allows CloudWatch Application Insights for .NET and SQL Server to complete the following actions on all resources:

- `cloudwatch:DescribeAlarmHistory`
- `cloudwatch:DescribeAlarms`
- `cloudwatch:GetMetricData`
- `cloudwatch:ListMetrics`
- `cloudwatch:PutMetricAlarm`
- `cloudwatch>DeleteAlarms`
- `logs:GetLogEvents`
- `logs:DescribeLogStreams`
- `logs:DescribeLogGroup`
- `events:DescribeRule`
- `tag:GetResources`
- `resource-groups:ListGroupResources`
- `resource-groups:GetGroupQuery`
- `elasticloadbalancing:DescribeLoadBalancers`
- `elasticloadbalancing:DescribeTargetGroups`

- elasticloadbalancing:DescribeTargetHealth
- Autoscaling:DescribeAutoScalingGroups
- cloudFormation:DescribeStacks
- cloudFormation:ListStackResources
- ssm:GetOpsItem
- ssm:CreateOpsItem
- ssm:DescribeOpsItems
- ssm:UpdateOpsItem
- ssm:DescribeInstanceInformation
- ec2:DescribeInstances
- rds:DescribeDBInstances

The role permissions policy allows CloudWatch Application Insights for .NET and SQL Server to complete the following actions on specified resources. These policies are used to manage resources created by or related to CloudWatch Application Insights for .NET and SQL Server in your account.

```
{
  "Effect": "Allow",
  "Action": [
    "cloudFormation:CreateStack",
    "cloudFormation:UpdateStack",
    "cloudFormation>DeleteStack"
  ],
  "Resource": [
    "arn:aws:cloudformation:*:*:stack/ApplicationInsights-*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "ssm:PutParameter",
    "ssm>DeleteParameter",
    "ssm:AddTagsToResource",
    "ssm:RemoveTagsFromResource"
  ],
  "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-ApplicationInsights-*"
},
{
  "Effect": "Allow",
  "Action": [
    "ssm>CreateAssociation",
    "ssm:UpdateAssociation",
    "ssm>DeleteAssociation",
    "ssm:DescribeAssociation"
  ],
  "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ssm:*:*:association/*",
    "arn:aws:ssm:*:*:managed-instance/*",
    "arn:aws:ssm:*:*:document/AWSEC2-ApplicationInsightsCloudwatchAgentInstallAndConfigure",
    "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
    "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
  ]
},
]
```

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see [Service-Linked Role Permissions](#) in the *IAM User Guide*.

Creating a Service-Linked Role for CloudWatch Application Insights for .NET and SQL Server

You don't need to manually create a service-linked role. When you create a new Application Insights application in the AWS Management Console, CloudWatch Application Insights for .NET and SQL Server creates the service-linked role for you.

If you delete this service-linked role, and then want to create it again, you can use the same process to recreate the role in your account. When you create a new Application Insights application, CloudWatch Application Insights for .NET and SQL Server creates the service-linked role for you again.

Editing a Service-Linked Role for CloudWatch Application Insights for .NET and SQL Server

CloudWatch Application Insights for .NET and SQL Server does not allow you to edit the ApplicationInsights-role service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see [Editing a Service-Linked Role](#) in the *IAM User Guide*.

Deleting a Service-Linked Role for CloudWatch Application Insights for .NET and SQL Server

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you avoid having an unused entity that is not actively monitored or maintained. However, you must delete all applications in Application Insights before you can manually delete the role.

Note

If the CloudWatch Application Insights for .NET and SQL Server service is using the role when you try to delete the resources, the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete CloudWatch Application Insights for .NET and SQL Server resources used by the ApplicationInsights-role

- Delete all of your CloudWatch Application Insights for .NET and SQL Server applications. For more information, see "Deleting Your Application(s)" in the CloudWatch Application Insights for .NET and SQL Server User Guide.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the ApplicationInsights-role service-linked role. For more information, see [Deleting a Service-Linked Role](#) in the *IAM User Guide*.

Supported Regions for CloudWatch Application Insights for .NET and SQL Server Service-Linked Roles

CloudWatch Application Insights for .NET and SQL Server supports using service-linked roles in all of the AWS Regions where the service is available.

Region name	Region identity	Support in CloudWatch Application Insights for .NET and SQL Server
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
Europe (Stockholm)	eu-north-1	Yes
South America (São Paulo)	sa-east-1	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Canada (Central)	ca-central-1	Yes

Amazon CloudWatch Permissions Reference

When you are setting up [Access Control](#) (p. 421) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can use the following table as a reference. The table lists each CloudWatch API operation and the corresponding actions for which you can grant permissions to perform the action. You specify the actions in the policy's `Action` field, and you specify a wildcard character (*) as the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your CloudWatch policies to express conditions. For a complete list of AWS-wide keys, see [AWS Global and IAM Condition Context Keys](#) in the *IAM User Guide*.

Note

To specify an action, use the `cloudwatch:` prefix followed by the API operation name. For example: `cloudwatch:GetMetricStatistics`, `cloudwatch:ListMetrics`, or `cloudwatch:*` (for all CloudWatch actions).

Tables

- [CloudWatch API Operations and Required Permissions](#)
- [CloudWatch Events API Operations and Required Permissions](#)
- [CloudWatch Logs API Operations and Required Permissions](#)
- [Amazon EC2 API Operations and Required Permissions](#)
- [Amazon EC2 Auto Scaling API Operations and Required Permissions](#)

CloudWatch API Operations and Required Permissions for Actions

CloudWatch API Operations	Required Permissions (API Actions)
DeleteAlarms	<code>cloudwatch:DeleteAlarms</code> Required to delete an alarm.
DeleteDashboards	<code>cloudwatch:DeleteDashboards</code> Required to delete a dashboard.
DescribeAlarmHistory	<code>cloudwatch:DescribeAlarmHistory</code> Required to view alarm history.
DescribeAlarms	<code>cloudwatch:DescribeAlarms</code> Required to retrieve alarm information by name.
DescribeAlarmsForMetric	<code>cloudwatch:DescribeAlarmsForMetric</code> Required to view alarms for a metric.
DisableAlarmActions	<code>cloudwatch:DisableAlarmActions</code> Required to disable an alarm action.
EnableAlarmActions	<code>cloudwatch:EnableAlarmActions</code> Required to enable an alarm action.
GetDashboard	<code>cloudwatch:GetDashboard</code> Required to display data about existing dashboards.
GetMetricData	<code>cloudwatch:GetMetricData</code> Required to retrieve large batches of metric data and perform metric math on that data.
GetMetricStatistics	<code>cloudwatch:GetMetricStatistics</code> Required to view graphs in other parts of the CloudWatch console and in dashboard widgets.
GetMetricWidgetImage	<code>cloudwatch:GetMetricWidgetImage</code> Required to retrieve a snapshot graph of one or more CloudWatch metrics as a bitmap image.
ListDashboards	<code>cloudwatch:ListDashboards</code> Required to view the list of CloudWatch dashboards in your account.
ListMetrics	<code>cloudwatch:ListMetrics</code> Required to view or search metric names within the CloudWatch console and in the CLI. Required to select metrics on dashboard widgets.

CloudWatch API Operations	Required Permissions (API Actions)
PutDashboard	<code>cloudwatch:PutDashboard</code> Required to create a dashboard or update an existing dashboard.
PutMetricAlarm	<code>cloudwatch:PutMetricAlarm</code> Required to create or update an alarm.
PutMetricData	<code>cloudwatch:PutMetricData</code> Required to create metrics.
SetAlarmState	<code>cloudwatch:SetAlarmState</code> Required to manually set an alarm's state.

CloudWatch Events API Operations and Required Permissions for Actions

CloudWatch Events API Operations	Required Permissions (API Actions)
DeleteRule	<code>events:DeleteRule</code> Required to delete a rule.
DescribeRule	<code>events:DescribeRule</code> Required to list the details about a rule.
DisableRule	<code>events:DisableRule</code> Required to disable a rule.
EnableRule	<code>events:EnableRule</code> Required to enable a rule.
ListRuleNamesByTarget	<code>events:ListRuleNamesByTarget</code> Required to list rules associated with a target.
ListRules	<code>events:ListRules</code> Required to list all rules in your account.
ListTargetsByRule	<code>events:ListTargetsByRule</code> Required to list all targets associated with a rule.
PutEvents	<code>events:PutEvents</code> Required to add custom events that can be matched to rules.
PutRule	<code>events:PutRule</code> Required to create or update a rule.
PutTargets	<code>events:PutTargets</code>

CloudWatch Events API Operations	Required Permissions (API Actions)
	Required to add targets to a rule.
RemoveTargets	<code>events:RemoveTargets</code> Required to remove a target from a rule.
TestEventPattern	<code>events:TestEventPattern</code> Required to test an event pattern against a given event.

CloudWatch Logs API Operations and Required Permissions for Actions

CloudWatch Logs API Operations	Required Permissions (API Actions)
CancelExportTask	<code>logs:CancelExportTask</code> Required to cancel a pending or running export task.
CreateExportTask	<code>logs:CreateExportTask</code> Required to export data from a log group to an Amazon S3 bucket.
CreateLogGroup	<code>logs:CreateLogGroup</code> Required to create a new log group.
CreateLogStream	<code>logs:CreateLogStream</code> Required to create a new log stream in a log group.
DeleteDestination	<code>logs:DeleteDestination</code> Required to delete a log destination and disables any subscription filters to it.
DeleteLogGroup	<code>logs:DeleteLogGroup</code> Required to delete a log group and any associated archived log events.
DeleteLogStream	<code>logs:DeleteLogStream</code> Required to delete a log stream and any associated archived log events.
DeleteMetricFilter	<code>logs:DeleteMetricFilter</code> Required to delete a metric filter associated with a log group.
DeleteQueryDefinition	<code>logs:DeleteQueryDefinition</code> Required to delete a saved query definition in CloudWatch Logs Insights.

CloudWatch Logs API Operations	Required Permissions (API Actions)
DeleteResourcePolicy	<code>logs:DeleteResourcePolicy</code> Required to delete a CloudWatch Logs resource policy.
DeleteRetentionPolicy	<code>logs:DeleteRetentionPolicy</code> Required to delete a log group's retention policy.
DeleteSubscriptionFilter	<code>logs:DeleteSubscriptionFilter</code> Required to delete the subscription filter associated with a log group.
DescribeDestinations	<code>logs:DescribeDestinations</code> Required to view all destinations associated with the account.
DescribeExportTasks	<code>logs:DescribeExportTasks</code> Required to view all export tasks associated with the account.
DescribeLogGroups	<code>logs:DescribeLogGroups</code> Required to view all log groups associated with the account.
DescribeLogStreams	<code>logs:DescribeLogStreams</code> Required to view all log streams associated with a log group.
DescribeMetricFilters	<code>logs:DescribeMetricFilters</code> Required to view all metrics associated with a log group.
DescribeQueryDefinitions	<code>logs:DescribeQueryDefinitions</code> Required to see the list of saved query definitions in CloudWatch Logs Insights.
DescribeQueries	<code>logs:DescribeQueries</code> Required to see the list of CloudWatch Logs Insights queries that are scheduled, executing, or have recently executed.
DescribeResourcePolicies	<code>logs:DescribeResourcePolicies</code> Required to view a list of CloudWatch Logs resource policies.
DescribeSubscriptionFilters	<code>logs:DescribeSubscriptionFilters</code> Required to view all subscription filters associated with a log group.

CloudWatch Logs API Operations	Required Permissions (API Actions)
FilterLogEvents	<code>logs:FilterLogEvents</code> Required to sort log events by log group filter pattern.
GetLogEvents	<code>logs:GetLogEvents</code> Required to retrieve log events from a log stream.
GetLogGroupFields	<code>logs:GetLogGroupFields</code> Required to retrieve the list of fields that are included in the log events in a log group.
GetLogRecord	<code>logs:GetLogRecord</code> Required to retrieve the details from a single log event.
GetQueryResults	<code>logs:GetQueryResults</code> Required to retrieve the results of CloudWatch Logs Insights queries.
ListTagsLogGroup	<code>logs:ListTagsLogGroup</code> Required to list the tags associated with a log group.
PutDestination	<code>logs:PutDestination</code> Required to create or update a destination log stream (such as an Kinesis stream).
PutDestinationPolicy	<code>logs:PutDestinationPolicy</code> Required to create or update an access policy associated with an existing log destination.
PutLogEvents	<code>logs:PutLogEvents</code> Required to upload a batch of log events to a log stream.
PutMetricFilter	<code>logs:PutMetricFilter</code> Required to create or update a metric filter and associate it with a log group.
PutQueryDefinition	<code>logs:PutQueryDefinition</code> Required to save a query in CloudWatch Logs Insights.
PutResourcePolicy	<code>logs:PutResourcePolicy</code> Required to create a CloudWatch Logs resource policy.

CloudWatch Logs API Operations	Required Permissions (API Actions)
PutRetentionPolicy	<code>logs:PutRetentionPolicy</code> Required to set the number of days to keep log events (retention) in a log group.
PutSubscriptionFilter	<code>logs:PutSubscriptionFilter</code> Required to create or update a subscription filter and associate it with a log group.
StartQuery	<code>logs:StartQuery</code> Required to start CloudWatch Logs Insights queries.
StopQuery	<code>logs:StopQuery</code> Required to stop a CloudWatch Logs Insights query that is in progress.
TagLogGroup	<code>logs:TagLogGroup</code> Required to add or update log group tags.
TestMetricFilter	<code>logs:TestMetricFilter</code> Required to test a filter pattern against a sampling of log event messages.

Amazon EC2 API Operations and Required Permissions for Actions

Amazon EC2 API Operations	Required Permissions (API Actions)
DescribeInstanceStatus	<code>ec2:DescribeInstanceStatus</code> Required to view EC2 instance status details.
DescribeInstances	<code>ec2:DescribeInstances</code> Required to view EC2 instance details.
RebootInstances	<code>ec2:RebootInstances</code> Required to reboot an EC2 instance.
StopInstances	<code>ec2:StopInstances</code> Required to stop an EC2 instance.
TerminateInstances	<code>ec2:TerminateInstances</code> Required to terminate an EC2 instance.

Amazon EC2 Auto Scaling API Operations and Required Permissions for Actions

Amazon EC2 Auto Scaling API Operations	Required Permissions (API Actions)
Scaling	<code>autoscaling:Scaling</code> Required to scale an Auto Scaling group.
Trigger	<code>autoscaling:Trigger</code> Required to trigger an Auto Scaling action.

Compliance Validation for Amazon CloudWatch

Third-party auditors assess the security and compliance of Amazon CloudWatch as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using Amazon CloudWatch is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and Compliance Quick Start Guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – AWS Config; assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon CloudWatch

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

Infrastructure Security in Amazon CloudWatch

As a managed service, Amazon CloudWatch is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access Amazon CloudWatch through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Network Isolation

A virtual private cloud (VPC) is a virtual network in your own logically isolated area in the AWS cloud. A subnet is a range of IP addresses in a VPC. You can deploy a variety of AWS resources in the subnets of your VPCs. For example, you can deploy Amazon EC2 instances, EMR clusters, and DynamoDB tables in subnets. For more information, see the [Amazon VPC User Guide](#).

To enable CloudWatch to communicate with resources in a VPC without going through the public internet, use AWS PrivateLink. For more information, see [Using CloudWatch and CloudWatch Synthetics with Interface VPC Endpoints](#) (p. 447).

A private subnet is a subnet with no default route to the public internet. Deploying an AWS resource in a private subnet does not prevent Amazon CloudWatch from collecting built-in metrics from the resource.

If you need to publish custom metrics from an AWS resource in a private subnet, you can do so using a proxy server. The proxy server forwards those HTTPS requests to the public API endpoints for CloudWatch.

Using CloudWatch and CloudWatch Synthetics with Interface VPC Endpoints

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC, CloudWatch, and CloudWatch Synthetics. You can use these connections to enable CloudWatch and CloudWatch Synthetics to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you can use to launch AWS resources in a virtual network that you define. With a VPC, you have control over your network settings, such as the IP address range, subnets, route tables, and network gateways. To connect your VPC to CloudWatch or CloudWatch Synthetics, you define an *interface VPC endpoint* to connect your VPC to AWS services. The endpoint provides reliable, scalable connectivity to CloudWatch or CloudWatch Synthetics without requiring an internet gateway, network address translation (NAT) instance, or VPN connection. For more information, see [What Is Amazon VPC](#) in the *Amazon VPC User Guide*.

Interface VPC endpoints are powered by AWS PrivateLink, an AWS technology that enables private communication between AWS services using an elastic network interface with private IP addresses. For more information, see the [New – AWS PrivateLink for AWS Services](#) blog post.

The following steps are for users of Amazon VPC. For more information, see [Getting Started](#) in the *Amazon VPC User Guide*.

CloudWatch VPC Endpoint

CloudWatch currently supports VPC endpoints in the following AWS Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- South America (São Paulo)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Creating a VPC Endpoint for CloudWatch

To start using CloudWatch with your VPC, create an interface VPC endpoint for CloudWatch. The service name to choose is `com.amazonaws.region.monitoring`. For more information, see [Creating an Interface Endpoint](#) in the *Amazon VPC User Guide*.

You do not need to change the settings for CloudWatch. CloudWatch calls other AWS services using either public endpoints or private interface VPC endpoints, whichever are in use. For example, if you create an interface VPC endpoint for CloudWatch, and you already have metrics flowing to CloudWatch from resources located on your VPC, these metrics begin flowing through the interface VPC endpoint by default.

Controlling Access to Your CloudWatch VPC Endpoint

A VPC endpoint policy is an IAM resource policy that you attach to an endpoint when you create or modify the endpoint. If you don't attach a policy when you create an endpoint, Amazon VPC attaches a default policy for you that allows full access to the service. An endpoint policy doesn't override or replace IAM user policies or service-specific policies. It's a separate policy for controlling access from the endpoint to the specified service.

Endpoint policies must be written in JSON format.

For more information, see [Controlling Access to Services with VPC Endpoints](#) in the *Amazon VPC User Guide*.

The following is an example of an endpoint policy for CloudWatch. This policy allows users connecting to CloudWatch through the VPC to send metric data to CloudWatch and prevents them from performing other CloudWatch actions.

```
{  
  "Statement": [  
    {  
      "Action": "cloudwatch:PutMetricData",  
      "Effect": "Allow",  
      "Resource": "*" }  
    ]  
}
```



```
{
  "Sid": "PutOnly",
  "Principal": "*",
  "Action": [
    "cloudwatch:PutMetricData"
  ],
  "Effect": "Allow",
  "Resource": "*"
}
```

To edit the VPC endpoint policy for CloudWatch

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints**.
3. If you have not already created the endpoint for CloudWatch, choose **Create Endpoint**. Select **com.amazonaws.*region*.monitoring**, and then choose **Create endpoint**.
4. Select the **com.amazonaws.*region*.monitoring** endpoint, and then choose the **Policy** tab.
5. Choose **Edit Policy**, and then make your changes.

CloudWatch Synthetics VPC Endpoint

CloudWatch Synthetics currently supports VPC endpoints in the following AWS Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- South America (São Paulo)

Creating a VPC Endpoint for CloudWatch Synthetics

To start using CloudWatch Synthetics with your VPC, create an interface VPC endpoint for CloudWatch Synthetics. The service name to choose is `com.amazonaws.region.synthetics`. For more information, see [Creating an Interface Endpoint](#) in the *Amazon VPC User Guide*.

You do not need to change the settings for CloudWatch Synthetics. CloudWatch Synthetics communicates with other AWS services using either public endpoints or private interface VPC

endpoints, whichever are in use. For example, if you create an interface VPC endpoint for CloudWatch Synthetics, and you already have an interface endpoint for Amazon S3, CloudWatch Synthetics begins communicating with Amazon S3 through the interface VPC endpoint by default.

Controlling Access to Your CloudWatch Synthetics VPC Endpoint

A VPC endpoint policy is an IAM resource policy that you attach to an endpoint when you create or modify the endpoint. If you don't attach a policy when you create an endpoint, we attach a default policy for you that allows full access to the service. An endpoint policy doesn't override or replace IAM user policies or service-specific policies. It's a separate policy for controlling access from the endpoint to the specified service.

Endpoint policies must be written in JSON format.

For more information, see [Controlling Access to Services with VPC Endpoints](#) in the *Amazon VPC User Guide*.

The following is an example of an endpoint policy for CloudWatch Synthetics. This policy enables users connecting to CloudWatch Synthetics through the VPC to view information about canaries and their runs, but not to create, modify, or delete canaries.

```
{
  "Statement": [
    {
      "Action": [
        "synthetics:DescribeCanaries",
        "synthetics:GetCanaryRuns"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

To edit the VPC endpoint policy for CloudWatch Synthetics

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints**.
3. If you have not already created the endpoint for CloudWatch Synthetics, choose **Create Endpoint**. Select **com.amazonaws.*region*.synthetics** and then choose **Create endpoint**.
4. Select the **com.amazonaws.*region*.synthetics** endpoint and then choose the **Policy** tab.
5. Choose **Edit Policy**, and then make your changes.

Security Considerations for Synthetics Canaries

The following sections explain security issues that you should consider when creating and running canaries in Synthetics.

Use Secure Connections

Because canary code and the results from canary test runs can contain sensitive information, do not have your canary connect to endpoints over unencrypted connections. Always use encrypted connections, such as those that begin with `https://`.

Canary Naming Considerations

The Amazon Resource Name (ARN) of a canary is included in the user-agent header as a part of outbound calls made from the Puppeteer-driven Chromium browser that is included as a part of the CloudWatch Synthetics wrapper library. This helps identify CloudWatch Synthetics canary traffic and relate it back to the canaries that are making calls.

The canary ARN includes the canary name. Choose canary names that do not reveal proprietary information.

Additionally, be sure to point your canaries only at websites and endpoints that you control.

Secrets in Canary Code

We recommend that you don't include secrets, such as access keys or database credentials, in your canary source code. For more information about how to use AWS Secrets Manager to help keep your secrets safe, see [What is AWS Secrets Manager?](#).

Permissions Considerations

We recommend that you restrict access to resources that are created or used by CloudWatch Synthetics. Use tight permissions on the Amazon S3 buckets where canaries store test run results and other artifacts, such as logs and screenshots.

Similarly, keep tight permissions on the locations where your canary source code is stored, so that no user accidentally or maliciously deletes the Lambda layers or Lambda functions used for the canary.

To help make sure you run the canary code you intend, you can use object versioning on the Amazon S3 bucket where your canary code is stored. Then when you specify this code to run as a canary, you can include the object `versionId` as part of the path, as in the following examples.

```
https://bucket.s3.amazonaws.com/path/object.zip?versionId=version-id  
https://s3.amazonaws.com/bucket/path/object.zip?versionId=version-id  
https://bucket.s3-region.amazonaws.com/path/object.zip?versionId=version-id
```

Stack Traces and Exception Messages

By default, CloudWatch Synthetics canaries capture any exception thrown by your canary script, no matter whether the script is custom or is from a blueprint. CloudWatch Synthetics logs both the exception message and the stack trace to three locations:

- Back into the CloudWatch Synthetics service to speed up debugging when you describe test runs
- Into CloudWatch Logs according to the configuration that your Lambda functions are created with
- Into the Synthetics log file, which is a plaintext file that is uploaded to the Amazon S3 location specified by the value you set for the `resultsLocation` of the canary

If you want to send and store less information, you can capture exceptions before they return to the CloudWatch Synthetics wrapper library.

Scope Your IAM Roles Narrowly

We recommend that you do not configure your canary to visit potentially malicious URLs or endpoints. Pointing your Canary to untrusted or unknown websites or endpoints could expose your Lambda

function code to malicious user's scripts. Assuming a malicious website can break out of Chromium, it could have access to your Lambda code in a similar way to if you connected to it using an internet browser.

Run your Lambda function with an IAM execution role that has scoped-down permissions. This way, if your Lambda function is compromised by a malicious script, it is limited in the actions it can take when running as your canary's AWS account.

When you use the CloudWatch console to create a canary, it is created with a scoped-down IAM execution role.

Header Logging

By default, canary header logging logs only the request URL, response code, and response status. Within your canary script, you can customize the logging to log more or less information.

Logging Amazon CloudWatch API Calls with AWS CloudTrail

Amazon CloudWatch and CloudWatch Synthetics are integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures API calls made by or on behalf of your AWS account. The captured calls include calls from the console and code calls to API operations.

If you create a *trail*, you can enable continuous delivery of CloudTrail events to an S3 bucket, including events for CloudWatch. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to CloudWatch, the IP address from which the request was made, who made the request, when it was made, and other details.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail `userIdentity` Element](#).

For an ongoing record of events in your AWS account, including events for CloudWatch and CloudWatch Synthetics, create a trail. A trail enables CloudTrail to deliver log files to an S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the S3 bucket that you specify. You can configure other AWS services to further analyze and act on the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

Topics

- [CloudWatch Information in CloudTrail \(p. 453\)](#)
- [CloudWatch Synthetics Information in CloudTrail \(p. 456\)](#)

CloudWatch Information in CloudTrail

CloudWatch supports logging the following actions as events in CloudTrail log files:

- [DeleteAlarms](#)
- [DeleteDashboards](#)
- [DescribeAlarmHistory](#)
- [DescribeAlarms](#)
- [DescribeAlarmsForMetric](#)
- [DisableAlarmActions](#)
- [EnableAlarmActions](#)
- [GetDashboard](#)
- [ListDashboards](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [SetAlarmState](#)

You can log the following AWS SDK Metrics actions in CloudTrail log files. These actions are used only by SDK Metrics and are not available for use in your code. For more information, see [Monitor Applications Using AWS SDK Metrics](#) (p. 402).

- `GetPublishingConfiguration` – Used to determine how to publish metrics.
- `GetPublishingSchema` – Used to determine how to publish metrics.
- `PutPublishingMetrics` – Used to send CloudWatch Agent health metrics to AWS Support.

Example: CloudWatch Log File Entries

The following example shows a CloudTrail log entry that demonstrates the `PutMetricAlarm` action.

```
{
  "Records": [{
    "eventVersion": "1.01",
    "userIdentity": {
      "type": "Root",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:root",
      "accountId": "123456789012",
      "accessKeyId": "EXAMPLE_KEY_ID"
    },
    "eventTime": "2014-03-23T21:50:34Z",
    "eventSource": "monitoring.amazonaws.com",
    "eventName": "PutMetricAlarm",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
    "requestParameters": {
      "threshold": 50.0,
      "period": 60,
      "metricName": "CloudTrail Test",
      "evaluationPeriods": 3,
      "comparisonOperator": "GreaterThanThreshold",
      "namespace": "AWS/CloudWatch",
      "alarmName": "CloudTrail Test Alarm",
      "statistic": "Sum"
    },
    "responseElements": null,
    "requestID": "29184022-b2d5-11e3-a63d-9b463e6d0ff0",
    "eventID": "b096d5b7-dcf2-4399-998b-5a53eca76a27"
  }],
  ..additional entries
}
```

```
]
}
```

The following log file entry shows that a user called the CloudWatch Events PutRule action.

```
{
  "eventVersion": "1.03",
  "userIdentity": {
    "type": "Root",
    "principalId": "123456789012",
    "arn": "arn:aws:iam::123456789012:root",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2015-11-17T23:56:15Z"
      }
    }
  },
  "eventTime": "2015-11-18T00:11:28Z",
  "eventSource": "events.amazonaws.com",
  "eventName": "PutRule",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS CloudWatch Console",
  "requestParameters": {
    "description": "",
    "name": "cttest2",
    "state": "ENABLED",
    "eventPattern": "{\"source\": [\"aws.ec2\"], \"detail-type\": [\"EC2 Instance State-change Notification\"]}",
    "scheduleExpression": ""
  },
  "responseElements": {
    "ruleArn": "arn:aws:events:us-east-1:123456789012:rule/cttest2"
  },
  "requestID": "e9caf887-8d88-11e5-a331-3332aa445952",
  "eventID": "49d14f36-6450-44a5-a501-b0fdcdfaeb98",
  "eventType": "AwsApiCall",
  "apiVersion": "2015-10-07",
  "recipientAccountId": "123456789012"
}
```

The following log file entry shows that a user called the CloudWatch Logs CreateExportTask action.

```
{
  "eventVersion": "1.03",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:user/someuser",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "someuser"
  },
  "eventTime": "2016-02-08T06:35:14Z",
  "eventSource": "logs.amazonaws.com",
  "eventName": "CreateExportTask",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
  "requestParameters": {

```

```
        "destination": "yourdestination",
        "logGroupName": "yourloggroup",
        "to": 123456789012,
        "from": 0,
        "taskName": "yourtask"
    },
    "responseElements": {
        "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
    },
    "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
    "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
    "eventType": "AwsApiCall",
    "apiVersion": "20140328",
    "recipientAccountId": "123456789012"
}
```

CloudWatch Synthetics Information in CloudTrail

CloudWatch Synthetics supports logging the following actions as events in CloudTrail log files:

- [CreateCanary](#)
- [DeleteCanary](#)
- [DescribeCanaries](#)
- [DescribeCanariesLastRun](#)
- [DescribeRuntimeVersions](#)
- [GetCanary](#)
- [GetCanaryRuns](#)
- [ListTagsForResource](#)
- [StartCanary](#)
- [StopCanary](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateCanary](#)

Example: CloudWatch Synthetics Log File Entries

The following example shows a CloudTrail Synthetics log entry that demonstrates the `DescribeCanaries` action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111222333444:role/Administrator",
        "accountId": "123456789012",
        "userName": "SAMPLE_NAME"
      }
    }
  }
}
```



```
    },
    "webIdFederationData": {},
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2020-04-08T21:43:24Z"
    }
  },
  "eventTime": "2020-04-08T23:06:47Z",
  "eventSource": "synthetics.amazonaws.com",
  "eventName": "DescribeCanaries",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "201ed5f3-15db-4f87-94a4-123456789",
  "eventID": "73ddb81-3dd0-4ada-b246-123456789",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "11112223333"
}
```

The following example shows a CloudTrail Synthetics log entry that demonstrates the `UpdateCanary` action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111222333444:role/Administrator",
        "accountId": "123456789012",
        "userName": "SAMPLE_NAME"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-04-08T21:43:24Z"
      }
    }
  },
  "eventTime": "2020-04-08T23:06:47Z",
  "eventSource": "synthetics.amazonaws.com",
  "eventName": "UpdateCanary",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
  "requestParameters": {
    "Schedule": {
      "Expression": "rate(1 minute)"
    },
    "name": "sample_canary_name",
  }
```

```
    "Code": {
      "Handler": "myOwnScript.handler",
      "ZipFile": "SAMPLE_ZIP_FILE"
    },
    "responseElements": null,
    "requestID": "fe4759b0-0849-4e0e-be71-1234567890",
    "eventID": "9dc60c83-c3c8-4fa5-bd02-1234567890",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }
}
```

The following example shows a CloudTrail Synthetics log entry that demonstrates the `GetCanaryRuns` action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111222333444:role/Administrator",
        "accountId": "123456789012",
        "userName": "SAMPLE_NAME"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-04-08T21:43:24Z"
      }
    }
  },
  "eventTime": "2020-04-08T23:06:30Z",
  "eventSource": "synthetics.amazonaws.com",
  "eventName": "GetCanaryRuns",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
  "requestParameters": {
    "Filter": "TIME_RANGE",
    "name": "sample_canary_name",
    "FilterValues": [
      "2020-04-08T23:00:00.000Z",
      "2020-04-08T23:10:00.000Z"
    ]
  },
  "responseElements": null,
  "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
  "eventID": "52723fd9-4a54-478c-ac55-1234567890",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Grafana Integration

You can use Grafana version 6.5.0 and later to contextually advance through the CloudWatch console and query a dynamic list of metrics by using wildcards. This can help you monitor metrics for AWS resources, such as Amazon Elastic Compute Cloud instances or containers. When new instances are created as part of an Auto Scaling event, they appear in the graph automatically. You don't need to track the new instance IDs. Prebuilt dashboards help simplify the getting started experience for monitoring Amazon EC2, Amazon Elastic Block Store, and AWS Lambda resources.

You can use Grafana version 7.0 and later to perform CloudWatch Logs Insights queries on log groups in CloudWatch Logs. You can visualize your query results in bar, line, and stacked graphs and in a table format. For more information about CloudWatch Logs Insights, see [Analyzing Log Data with CloudWatch Logs Insights](#).

For more information about how to get started, see [Using AWS CloudWatch in Grafana](#) in the Grafana Labs documentation.

CloudWatch Service Quotas

CloudWatch has the following quotas for metrics, alarms, API requests, email notifications, and logs.

Resource	Default Quota
Actions	5/alarm. This quota cannot be changed.
Alarms	<p>10/month/customer for free. 5000 per Region, per account. You can request a quota increase.</p> <p>Alarms based on metric math expressions can have up to 10 metrics.</p>
Anomaly detection models	500 per Region, per account.
API requests	1,000,000/month/customer for free.
Canaries	100 per Region per account in the following Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), Europe (Ireland), and Asia Pacific (Tokyo). 20 per Region per account in all other Regions.
Contributor Insights API requests	<p>GetInsightRuleReport has a quota of 20 transactions per second (TPS). You can request a quota increase.</p> <p>The following APIs have a quota of 1 transaction per second. This quota cannot be changed.</p> <ul style="list-style-type: none"> • DeleteInsightRules • DescribeInsightRules • DisableInsightRules • EnableInsightRules • PutInsightRule
Contributor Insights rules	<p>100 rules per account.</p> <p>You can request a quota increase.</p>
Custom metrics	No quota.
Dashboards	<p>Up to 500 metrics per dashboard widget. Up to 2500 metrics per dashboard, across all widgets.</p> <p>These quotas include all metrics retrieved for use in metric math functions, even if those metrics are not displayed on the graph.</p> <p>These quotas cannot be changed.</p>
DescribeAlarms	9 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled.

Resource	Default Quota
	You can request a quota increase .
DeleteAlarms request DescribeAlarmHistory request DescribeAlarmsForMetric request DisableAlarmActions request EnableAlarmActions request SetAlarmState request	<p>3 transactions per second (TPS) for each of these operations. The maximum number of operation requests you can make per second without being throttled.</p> <p>These quotas cannot be changed.</p>
DeleteDashboards request GetDashboard request ListDashboards request PutDashboard request	<p>10 transactions per second (TPS) for each of these operations. The maximum number of operation requests you can make per second without being throttled.</p> <p>These quotas cannot be changed.</p>
PutAnomalyDetector DescribeAnomalyDetectors	10 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled.
DeleteAnomalyDetector	5 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled.
Dimensions	10/metric. This quota cannot be changed.
GetMetricData	<p>50 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a quota increase.</p> <p>180,000 Datapoints Per Second (DPS) if the <code>StartTime</code> used in the API request is less than or equal to three hours from current time. 396,000 DPS if the <code>StartTime</code> is more than three hours from current time. This is the maximum number of datapoints you can request per second using one or more API calls without being throttled. This quota cannot be changed.</p> <p>The DPS is calculated based on estimated data points, not actual data points. The data point estimate is calculated using the requested time range, period, and retention period. This means that if the actual data points in the requested metrics are sparse or empty, throttling still occurs if the estimated data points exceed the quota.</p>
GetMetricData	<p>A single <code>GetMetricData</code> call can include as many as 500 <code>MetricDataQuery</code> structures.</p> <p>This quota cannot be changed.</p>

Resource	Default Quota
GetMetricStatistics	400 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a quota increase .
GetMetricWidgetImage	Up to 500 metrics per image. This quota cannot be changed. 20 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. This quota cannot be changed.
ListMetrics	25 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a quota increase .
Metric data storage	15 months. This quota cannot be changed.
Metric data values	The value of a metric data point must be within the range of -2^{360} to 2^{360} . Special values (for example, NaN, +Infinity, -Infinity) are not supported. This quota cannot be changed.
Metrics	10/month/customer for free.
MetricDatum items	20/ PutMetricData request. A MetricDatum object can contain a single value or a StatisticSet object representing many values. This quota cannot be changed.
Metrics	10/month/customer for free.
Period	Maximum value is one day (86,400 seconds). This quota cannot be changed.
PutMetricAlarm request	3 transactions per second (TPS). The maximum number of operation requests you can make per second without being throttled. You can request a quota increase .
PutMetricData request	40 KB for HTTP POST requests. PutMetricData can handle 150 transactions per second (TPS), which is the maximum number of operation requests you can make per second without being throttled. You can request a quota increase .
Amazon SNS email notifications	1,000/month/customer for free.

Document History

The following table describes important changes in each release of the *Amazon CloudWatch User Guide*, beginning in June 2018. For notification about updates to this documentation, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
CloudWatch Contributor Insights general availability (p. 463)	CloudWatch Contributor Insights is now generally available. It enables you to analyze log data and create time series that display contributor data. You can see metrics about the top-N contributors, the total number of unique contributors, and their usage. For more information, see Using Contributor Insights to Analyze High-Cardinality Data in the <i>Amazon CloudWatch User Guide</i> .	April 2, 2020
CloudWatch Synthetics public preview (p. 463)	CloudWatch Synthetics is now in public preview. It enables you to create canaries to monitor your endpoints and APIs. For more information, see Using Canaries in the <i>Amazon CloudWatch User Guide</i> .	November 25, 2019
CloudWatch Contributor Insights public preview (p. 463)	CloudWatch Contributor Insights is now in public preview. It enables you to analyze log data and create time series that display contributor data. You can see metrics about the top-N contributors, the total number of unique contributors, and their usage. For more information, see Using Contributor Insights to Analyze High-Cardinality Data in the <i>Amazon CloudWatch User Guide</i> .	November 25, 2019
CloudWatch launches ServiceLens feature (p. 463)	ServiceLens enhances the observability of your services and applications by enabling you to integrate traces, metrics, logs, and alarms into one place. ServiceLens integrates CloudWatch with AWS X-Ray to provide an end-to-end view of your application. For more information, see Using ServiceLens to Monitor the	November 21, 2019

	Health of Your Applications in the <i>Amazon CloudWatch User Guide</i> .	
Use CloudWatch to proactively manage your AWS service quotas (p. 463)	You can use CloudWatch to proactively manage your AWS service quotas. CloudWatch usage metrics provide visibility into your account's usage of resources and API operations. For more information, see Service Quotas Integration and Usage Metrics in the <i>Amazon CloudWatch User Guide</i> .	November 19, 2019
CloudWatch sends events when alarms change state (p. 463)	CloudWatch now sends an event to Amazon EventBridge when any CloudWatch alarm changes state. For more information, see Alarm Events and EventBridge in the <i>Amazon CloudWatch User Guide</i> .	October 8, 2019
Container Insights (p. 463)	CloudWatch Container Insights is now generally available. It enables you to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. For more information, see Using Container Insights in the <i>Amazon CloudWatch User Guide</i> .	August 30, 2019
Updates for Container Insights preview metrics on Amazon EKS and Kubernetes (p. 463)	The Container Insights on Amazon EKS and Kubernetes public preview has been updated. InstanceId is now included as a dimension to the cluster EC2 instances. This allows alarms that have been created on these metrics to trigger the following EC2 actions: Stop, Terminate, Reboot, or Recover. Additionally, pod and service metrics are now reported by Kubernetes namespace to simplify the monitoring and alarming on metrics by namespace.	August 19, 2019
Updates for AWS Systems Manager OpsCenter integration (p. 463)	Updates on how CloudWatch Application Insights integrates with Systems Manager OpsCenter.	August 7, 2019

CloudWatch usage metrics (p. 463)	CloudWatch usage metrics help you track the usage of your CloudWatch resources and stay within your service limits. For more information, see https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Usage-Metrics.html .	August 6, 2019
CloudWatch Container Insights public preview (p. 463)	CloudWatch Container Insights is now in public preview. It enables you to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. For more information, see Using Container Insights in the <i>Amazon CloudWatch User Guide</i> .	July 9, 2019
CloudWatch Anomaly Detection public preview (p. 463)	CloudWatch anomaly detection is now in public preview. CloudWatch applies machine-learning algorithms to a metric's past data to create a model of the metric's expected values. You can use this model for visualization and for setting alarms. For more information, see Using CloudWatch Anomaly Detection in the <i>Amazon CloudWatch User Guide</i> .	July 9, 2019
CloudWatch Application Insights for .NET and SQL Server (p. 463)	CloudWatch Application Insights for .NET and SQL Server facilitates observability for .NET and SQL Server applications. It can help you set up the best monitors for your application resources to continuously analyze data for signs of problems with your applications.	June 21, 2019
CloudWatch agent section re-organized (p. 463)	The CloudWatch agent documentation has been rewritten to improve clarity, especially for customers using the command line to install and configure the agent. For more information, see Collecting Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent in the <i>Amazon CloudWatch User Guide</i> .	March 28, 2019

SEARCH function added to metric math expressions (p. 463)	You can now use a SEARCH function in metric math expressions. This enables you to create dashboards that update automatically as new resources are created that match the search query. For more information, see Using Search Expressions in Graphs in the <i>Amazon CloudWatch User Guide</i> .	March 21, 2019
AWS SDK Metrics for Enterprise Support (p. 463)	SDK Metrics helps you assess the health of your AWS services and diagnose latency caused by reaching your account usage limits or by a service outage. For more information, see Monitor Applications Using AWS SDK Metrics in the <i>Amazon CloudWatch User Guide</i> .	December 11, 2018
Alarms on math expressions (p. 463)	CloudWatch supports creating alarms based on metric math expressions. For more information, see Alarms on Math Expressions in the <i>Amazon CloudWatch User Guide</i> .	November 20, 2018
New CloudWatch console home page (p. 463)	Amazon has created a new home page in the CloudWatch console, which automatically displays key metrics and alarms for all the AWS services you are using. For more information, see Getting Started with Amazon CloudWatch in the <i>Amazon CloudWatch User Guide</i> .	November 19, 2018
AWS CloudFormation templates for the CloudWatch Agent (p. 463)	Amazon has uploaded AWS CloudFormation templates that you can use to install and update the CloudWatch agent. For more information, see Install the CloudWatch Agent on New Instances Using AWS CloudFormation in the <i>Amazon CloudWatch User Guide</i> .	November 9, 2018

Enhancements to the CloudWatch Agent (p. 463)	The CloudWatch agent has been updated to work with both the StatsD and collectd protocols. It also has improved cross-account support. For more information, see Retrieve Custom Metrics with StatsD , Retrieve Custom Metrics with collectd , and Sending Metrics and Logs to a Different AWS Account in the <i>Amazon CloudWatch User Guide</i> .	September 28, 2018
Support for Amazon VPC endpoints (p. 463)	You can now establish a private connection between your VPC and CloudWatch. For more information, see Using CloudWatch with Interface VPC Endpoints in the <i>Amazon CloudWatch User Guide</i> .	June 28, 2018

The following table describes important changes to the *Amazon CloudWatch User Guide* before June 2018.

Change	Description	Release Date
Metric math	You can now perform math expressions on CloudWatch metrics, producing new time series that you can add to graphs on your dashboard. For more information, see Using Metric Math (p. 52) .	April 4, 2018
"M out of N" alarms	You can now configure an alarm to trigger based on "M out of N" datapoints in any alarm evaluation interval. For more information, see Evaluating an Alarm (p. 72) .	December 8, 2017
CloudWatch agent	A new unified CloudWatch agent was released. You can use the unified multi-platform agent to collect custom both system metrics and log files from Amazon EC2 instances and on-premises servers. The new agent supports both Windows and Linux and enables customization of metrics collected, including sub-resource metrics such as per-CPU core. For more information, see Collecting Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CloudWatch Agent (p. 236) .	September 7, 2017
NAT gateway metrics	Added metrics for Amazon VPC NAT gateway.	September 7, 2017
High-resolution metrics	You can now optionally set up custom metrics as high-resolution metrics, with a granularity of as low as one second. For more information, see High-Resolution Metrics (p. 50) .	July 26, 2017

Change	Description	Release Date
Dashboard APIs	You can now create, modify, and delete dashboards using APIs and the AWS CLI. For more information, see Creating a CloudWatch Dashboard (p. 17) .	July 6, 2017
AWS Direct Connect metrics	Added metrics for AWS Direct Connect.	June 29, 2017
Amazon VPC VPN metrics	Added metrics for Amazon VPC VPN.	May 15, 2017
AppStream 2.0 metrics	Added metrics for AppStream 2.0.	March 8, 2017
CloudWatch console color picker	You can now choose the color for each metric on your dashboard widgets. For more information, see Edit a Graph on a CloudWatch Dashboard (p. 23) .	February 27, 2017
Alarms on dashboards	Alarms can now be added to dashboards. For more information, see Add or Remove an Alarm from a CloudWatch Dashboard (p. 26) .	February 15, 2017
Added metrics for Amazon Polly	Added metrics for Amazon Polly.	December 1, 2016
Added metrics for Amazon Kinesis Data Analytics	Added metrics for Amazon Kinesis Data Analytics.	December 1, 2016
Added support for percentile statistics	You can specify any percentile, using up to two decimal places (for example, p95.45). For more information, see Percentiles (p. 7) .	November 17, 2016
Added metrics for Amazon Simple Email Service	Added metrics for Amazon Simple Email Service.	November 2, 2016
Updated metrics retention	Amazon CloudWatch now retains metrics data for 15 months instead of 14 days.	November 1, 2016
Updated metrics console interface	The CloudWatch console is updated with improvements to existing functionality and new functionality.	November 1, 2016
Added metrics for Amazon Elastic Transcoder	Added metrics for Amazon Elastic Transcoder.	September 20, 2016
Added metrics for Amazon API Gateway	Added metrics for Amazon API Gateway.	September 9, 2016
Added metrics for AWS Key Management Service	Added metrics for AWS Key Management Service.	September 9, 2016

Change	Description	Release Date
Added metrics for the new Application Load Balancers supported by Elastic Load Balancing	Added metrics for Application Load Balancers.	August 11, 2016
Added new NetworkPacketsIn and NetworkPacketsOut metrics for Amazon EC2	Added new NetworkPacketsIn and NetworkPacketsOut metrics for Amazon EC2.	March 23, 2016
Added new metrics for Amazon EC2 Spot fleet	Added new metrics for Amazon EC2 Spot fleet.	March 21, 2016
Added new CloudWatch Logs metrics	Added new CloudWatch Logs metrics.	March 10, 2016
Added Amazon Elasticsearch Service and AWS WAF metrics and dimensions	Added Amazon Elasticsearch Service and AWS WAF metrics and dimensions.	October 14, 2015
Added support for CloudWatch dashboards	Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those that are spread out across different regions. For more information, see Using Amazon CloudWatch Dashboards (p. 16) .	October 8, 2015
Added AWS Lambda metrics and dimensions	Added AWS Lambda metrics and dimensions.	September 4, 2015
Added Amazon Elastic Container Service metrics and dimensions	Added Amazon Elastic Container Service metrics and dimensions.	August 17, 2015
Added Amazon Simple Storage Service metrics and dimensions	Added Amazon Simple Storage Service metrics and dimensions.	July 26, 2015
New feature: Reboot alarm action	Added the reboot alarm action and new IAM role for use with alarm actions. For more information, see Create Alarms to Stop, Terminate, Reboot, or Recover an Instance (p. 95) .	July 23, 2015

Change	Description	Release Date
Added Amazon WorkSpaces metrics and dimensions	Added Amazon WorkSpaces metrics and dimensions.	April 30, 2015
Added Amazon Machine Learning metrics and dimensions	Added Amazon Machine Learning metrics and dimensions.	April 9, 2015
New feature: Amazon EC2 instance recovery alarm actions	Updated alarm actions to include new EC2 instance recovery action. For more information, see Create Alarms to Stop, Terminate, Reboot, or Recover an Instance (p. 95).	March 12, 2015
Added Amazon CloudFront and Amazon CloudSearch metrics and dimensions	Added Amazon CloudFront and Amazon CloudSearch metrics and dimensions.	March 6, 2015
Added Amazon Simple Workflow Service metrics and dimensions	Added Amazon Simple Workflow Service metrics and dimensions.	May 9, 2014
Updated guide to add support for AWS CloudTrail	Added a new topic to explain how you can use AWS CloudTrail to log activity in Amazon CloudWatch. For more information, see Logging Amazon CloudWatch API Calls with AWS CloudTrail (p. 453).	April 30, 2014
Updated guide to use the new AWS Command Line Interface (AWS 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>	February 21, 2014
Added Amazon Redshift and AWS OpsWorks metrics and dimensions	Added Amazon Redshift and AWS OpsWorks metrics and dimensions.	July 16, 2013
Added Amazon Route 53 metrics and dimensions	Added Amazon Route 53 metrics and dimensions.	June 26, 2013

Change	Description	Release Date
New feature: Amazon CloudWatch Alarm Actions	Added a new section to document Amazon CloudWatch alarm actions, which you can use to stop or terminate an Amazon Elastic Compute Cloud instance. For more information, see Create Alarms to Stop, Terminate, Reboot, or Recover an Instance (p. 95).	January 8, 2013
Updated EBS metrics	Updated the EBS metrics to include two new metrics for Provisioned IOPS volumes.	November 20, 2012
New billing alerts	You can now monitor your AWS charges using Amazon CloudWatch metrics and create alarms to notify you when you have exceeded the specified threshold. For more information, see Creating a Billing Alarm to Monitor Your Estimated AWS Charges (p. 100).	May 10, 2012
New metrics	You can now access six new Elastic Load Balancing metrics that provide counts of various HTTP response codes.	October 19, 2011
New feature	You can now access metrics from Amazon EMR.	June 30, 2011
New feature	You can now access metrics from Amazon Simple Notification Service and Amazon Simple Queue Service.	July 14, 2011
New Feature	Added information about using the <code>PutMetricData</code> API to publish custom metrics. For more information, see Publishing Custom Metrics (p. 50).	May 10, 2011
Updated metrics retention	Amazon CloudWatch now retains the history of an alarm for two weeks rather than six weeks. With this change, the retention period for alarms matches the retention period for metrics data.	April 7, 2011
New feature	Added ability to send Amazon Simple Notification Service or Auto Scaling notifications when a metric has crossed a threshold. For more information, see Alarms (p. 7).	December 2, 2010
New feature	A number of CloudWatch actions now include the <code>MaxRecords</code> and <code>NextToken</code> parameters, which enable you to control pages of results to display.	December 2, 2010
New feature	This service now integrates with AWS Identity and Access Management (IAM).	December 2, 2010