Amazon CloudWatch

Developer Guide

API Version 2010-08-01

Amazon Web Services

Amazon CloudWatch: Developer Guide

Amazon Web Services

Welcome

This is the *Amazon CloudWatch Developer Guide*. This guide contains conceptual information about the Amazon CloudWatch web service, as well as information about how to use the service to create new web applications or integrate with existing ones. Separate sections describe not only how to program with the Command Line Interface (CLI) and the Query API, but also how to integrate Amazon CloudWatch with other Amazon Web Services, such as Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing, Amazon Simple Notification Service (Amazon SNS), and Auto Scaling.

Amazon CloudWatch is a web service that enables you to monitor, manage, and publish various metrics, as well as configure alarm actions based on data from metrics.

How Do I...?

How Do I...	Relevant Topics
Get a general product overview and information about pricing	Amazon CloudWatch product information
Get a quick hands-on introduction to Amazon CloudWatch	Amazon CloudWatch Getting Started Guide
Get a quick summary of how Amazon CloudWatch works	Introduction to Amazon CloudWatch
Find available libraries for programmatically accessing Amazon CloudWatch	Available Libraries (p. 22)
Get started using the command line tools	Command Line Tools (p. 13)
Get started using the Query API for EC2	Query API (p. 19)

Introduction to Amazon CloudWatch

Topics

What Is Amazon CloudWatch?

Amazon CloudWatch is a web service that enables you to monitor, manage, and publish various metrics, as well as configure alarm actions based on data from metrics.

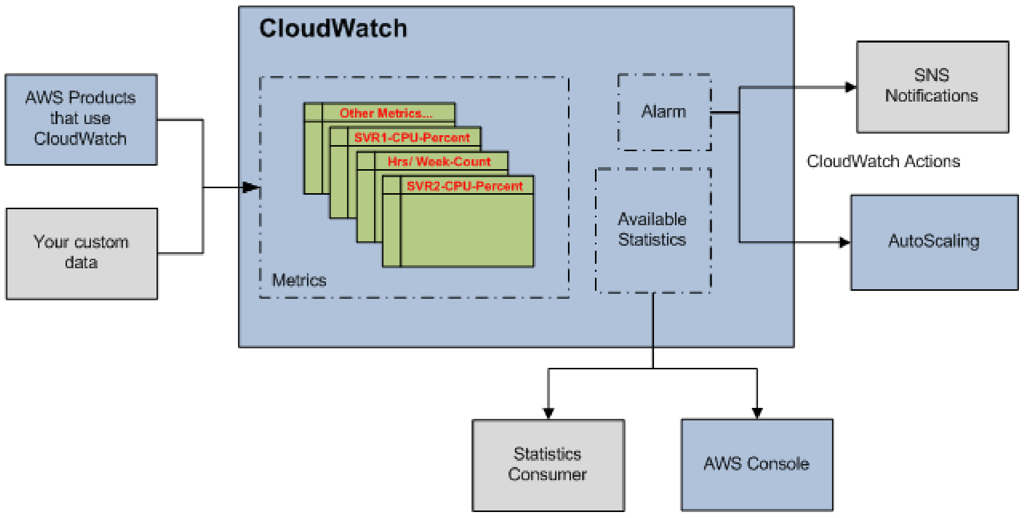
Amazon CloudWatch monitoring enables you to collect, analyze, and view system and application metrics so that you can make operational and business decisions more quickly and with greater confidence. You can use Amazon CloudWatch to collect metrics about your Amazon Web Services (AWS) resources—such as the performance of your Amazon EC2 instances. You can also publish your own metrics directly to Amazon CloudWatch.

Amazon CloudWatch allows you to manage the metrics in several ways. If you are publishing your own metrics, you can define custom metrics for your own use. If you are registered for an AWS product that supports Amazon CloudWatch, the service automatically pushes metrics to CloudWatch for you. Once Amazon CloudWatch contains either your custom metrics or metrics from an AWS service, you can calculate statistics based on that data and graphically visualize those statistics in the Amazon CloudWatch console. For more information about AWS products that Amazon CloudWatch supports, see CloudWatch Metrics, Namespaces, and Dimensions Reference (p. 36).

Amazon CloudWatch alarms help you implement decisions more easily by enabling you to send notifications or automatically make changes to the resources you are monitoring, based on rules that you define. For example, you can create alarms that initiate Auto Scaling and Simple Notification Service actions on your behalf. For more information about alarms, see Alarms (p. 9).

A common use for Amazon CloudWatch is to keep your applications and services healthy and running efficiently. For example, you can use it to discover that your website runs best when network traffic remains below a certain threshold level on your Amazon EC2 instances. You can then create an automated procedure to ensure that you always have the right number of instances to match the amount of traffic you have. You can also use Amazon CloudWatch to diagnose problems by looking at system performance before and after a problem occurs. Amazon CloudWatch helps you identify the cause and verify your fix by tracking performance in real time. For example, you can set up Amazon CloudWatch to email you right away when your application slows down, go back and discover that a particular database was being overloaded, and later watch response times come back up to speed.

As you can see in the following figure, Amazon CloudWatch is basically a metrics repository. An AWS product—such as Amazon EC2—puts metrics into the repository, and you retrieve statistics based on those metrics. If you put your own custom metrics into the repository, you can retrieve statistics based on your custom metrics.



Amazon CloudWatch Concepts

Topics

This section describes the terminology and concepts that are central to your understanding and use of Amazon CloudWatch.

Metrics

A *metric* is the fundamental concept for Amazon CloudWatch and represents a time-ordered set of data points. Either you or AWS products publish metric data points into Amazon CloudWatch and you retrieve statistics about those data points as an ordered set of time-series data.

You can think of a metric as a variable that you want to monitor. The data points represent the values of that variable over time. For example, the CPU usage of a particular Amazon EC2 instance is one metric, and the latency of an elastic load balancer is another.

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

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

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

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

Related Topics

- `PutMetricData` (mon-put-data Command (p. 115))
- `ListMetrics` (mon-list-metrics Command (p. 113))
- `GetMetricStatistics` (mon-get-stats Command (p. 110))
- Viewing Your AWS Metrics With CloudWatch (p. 24)

Namespaces

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

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

> **Note**
>
> Namespace names must be fewer than 256 characters in length.

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

Related Topics

- AWS Namespaces (p. 37)
- Getting Statistics Aggregated Across All Instances (p. 27)

Dimensions

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

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

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

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

Note

You can assign up to ten dimensions to a metric.

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

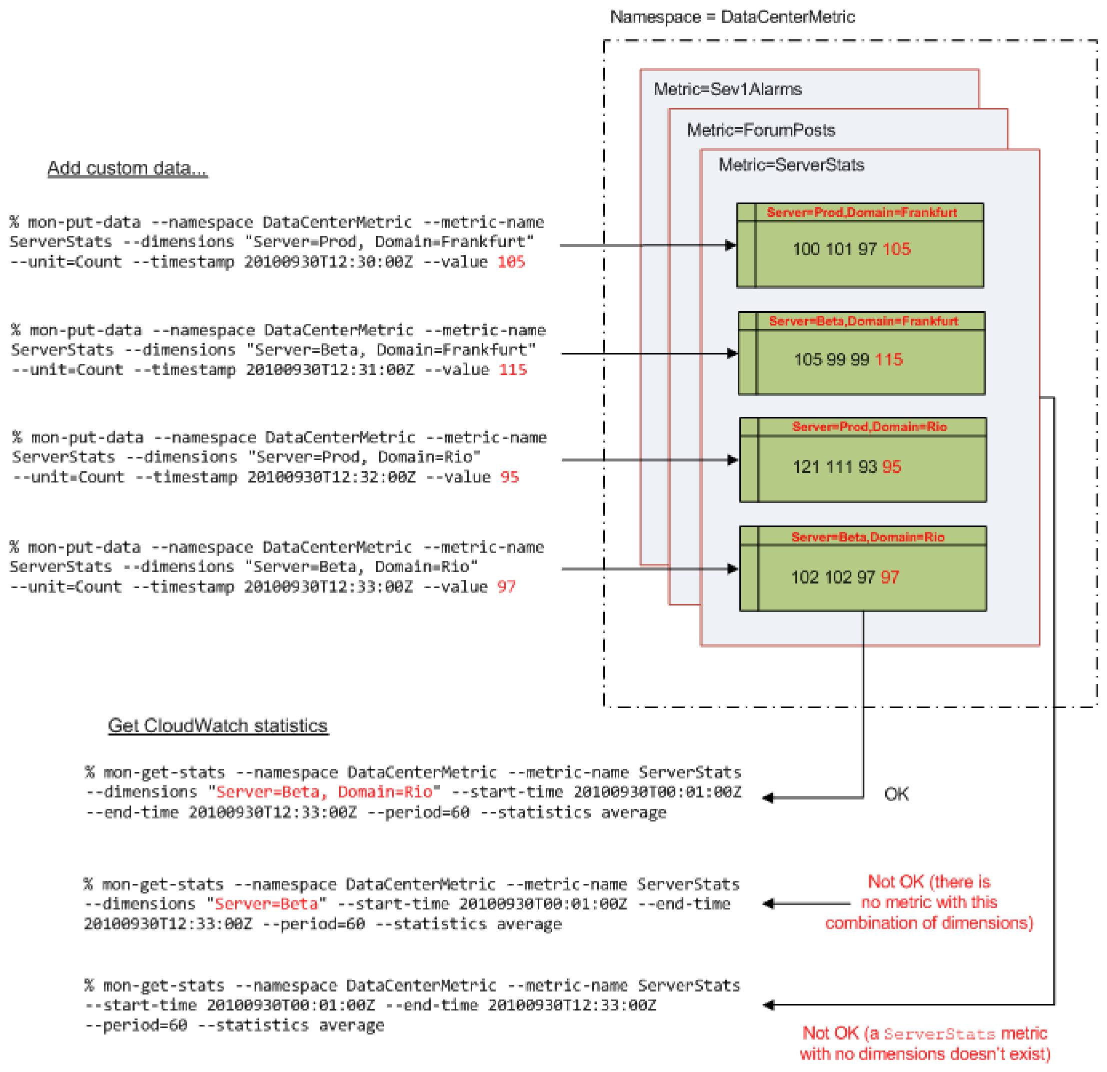
- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`
- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

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

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

Important

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



Add custom data...

```
% mon-put-data --namespace DataCenterMetric --metric-name
ServerStats --dimensions "Server=Prod, Domain=Frankfurt"
--unit=Count --timestamp 20100930T12:30:00Z --value 105
```

```
% mon-put-data --namespace DataCenterMetric --metric-name
ServerStats --dimensions "Server=Beta, Domain=Frankfurt"
--unit=Count --timestamp 20100930T12:31:00Z --value 115
```

```
% mon-put-data --namespace DataCenterMetric --metric-name
ServerStats --dimensions "Server=Prod, Domain=Rio"
--unit=Count --timestamp 20100930T12:32:00Z --value 95
```

```
% mon-put-data --namespace DataCenterMetric --metric-name
ServerStats --dimensions "Server=Beta, Domain=Rio"
--unit=Count --timestamp 20100930T12:33:00Z --value 97
```

Get CloudWatch statistics

```
% mon-get-stats --namespace DataCenterMetric --metric-name ServerStats
--dimensions "Server=Beta, Domain=Rio" --start-time 20100930T00:01:00Z
--end-time 20100930T12:33:00Z --period=60 --statistics average
```

OK

```
% mon-get-stats --namespace DataCenterMetric --metric-name ServerStats
--dimensions "Server=Beta" --start-time 20100930T00:01:00Z --end-time
20100930T12:33:00Z --period=60 --statistics average
```

Not OK (there is no metric with this combination of dimensions)

```
% mon-get-stats --namespace DataCenterMetric --metric-name ServerStats
--start-time 20100930T00:01:00Z --end-time 20100930T12:33:00Z
--period=60 --statistics average
```

Not OK (a `ServerStats` metric with no dimensions doesn't exist)

Related Topics

- mon-put-data Command (p. 115)
- mon-get-stats Command (p. 110)
- Dimensions for Amazon EC2 Metrics (p. 47)
- Dimensions for Elastic Load Balancing Metrics (p. 64)
- Dimensions for RDS Metrics (p. 54)

Time stamps

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

The time stamp you use in the request must be a `dateTime` object, with the complete date plus hours, minutes, and seconds (for more information, go to http://www.w3.org/TR/xmlschema-2/#dateTime). For example: 2007-01-31T23:59:59Z. Although it is not required, we recommend you provide the time stamp in the Coordinated Universal Time (UTC or Greenwich Mean Time) time zone. When you retrieve your statistics from Amazon CloudWatch, all times reflect the UTC time zone.

Units

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

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

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

For a complete list, go to the MetricDatum data type in the Amazon CloudWatch API Reference.

Though Amazon CloudWatch attaches no significance to a unit internally, other applications can derive semantic information based on the unit you choose. When you publish data without specifying a unit, Amazon CloudWatch associates it with the `None` unit. When you get statistics without specifying a unit, Amazon CloudWatch aggregates all data points of the same unit together. If you have two otherwise identical metrics with different units, two separate data streams will be returned, one for each unit.

Statistics

Statistics are metric data aggregations over specified periods of time. Amazon CloudWatch provides statistics based on the metric data points you or AWS products have provided to Amazon 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.

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

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

Amazon CloudWatch allows you to add pre-calculated statistics using the `PutMetricData` API (`mon-put-data` command) using the *`StatisticValues`* (*`statistic-values`*) parameter. Instead of data point values, you specify values for `SampleCount`, `Minimum`, `Maximum`, and `Sum` (Amazon CloudWatch calculates the average for you). The values you add in this way are aggregated with any other values associated with the matching metric.

Related Topics

- `PutMetricData` (mon-put-data Command (p. 115))
- `GetMetricStatistics` (mon-get-stats Command (p. 110))

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. You can adjust how the data is aggregated by varying the length of the period. A period can be as short as one minute (60 seconds) or as long as two weeks (1,209,600 seconds).

Note

Although periods are expressed in seconds, the minimum granularity for a period is one minute. Accordingly, you specify period values as multiples of 60. For example, to specify a period of six minutes, you would use the value 360.

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

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

Periods are also an important part of the Amazon CloudWatch Alarms feature. When you create an alarm to monitor a specific metric, you are asking Amazon CloudWatch to compare that metric to the threshold value that you supplied. You have extensive control over how Amazon CloudWatch makes that comparison. Not only can you specify the period over which the comparison is made, but you can also specify how many consecutive periods the threshold must be breached before you are notified. For more information on Alarms, see Alarms (p. 9).

Aggregation

Amazon CloudWatch aggregates statistics according to the period length that you specify in calls to `GetMetricStatistics`. You can publish as many data points as you want with the same or similar time stamps. Amazon CloudWatch aggregates them by period length when you get statistics about those data points with `GetMetricStatistics`.

You can publish data points for a metric that share not only the same time stamp, but also the same namespace and dimensions. Subsequent calls to `GetMetricStatistics` will return aggregated statistics about those data points. You can even do this in one `PutMetricData` request. Amazon CloudWatch accepts multiple data points in the same `PutMetricData` call with the same time stamp. You can also publish multiple data points for the same or different metrics, with any time stamp. The size of a `PutMetricData` request, however, is limited to 8KB for HTTP GET requests and 40KB for HTTP POST requests. You can include a maximum of 20 data points in one `PutMetricData` request.

For large data sets that would make the use of `PutMetricData` impractical, Amazon CloudWatch allows for the insertion of a pre-aggregated data set called a *StatisticSet*. With StatisticSets you give Amazon CloudWatch the Min, Max, Sum, and SampleCount of a number of datapoints. A common use case for StatisticSets is when you are collecting data many times in a minute. For example, let’s say you have a metric for the request latency of a web page. It doesn’t make sense to do a `PutMetricData` request with every web page hit. We suggest you collect the latency of all hits to that web page, aggregate them together once a minute and send that StatisticSet to Amazon 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, Amazon 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. Amazon CloudWatch treats these as a single metric, allowing you to get the statistics for minimum, maximum, average, and sum of all requests across your application.

Alarms

Amazon CloudWatch is especially useful because it helps you make decisions and take immediate, automatic actions based on your metric data. Alarms can automatically initiate actions on your behalf, based on parameters you specify. An alarm watches a single metric over a time period you specify, and performs one or more *actions* based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or Auto Scaling policy. Alarms invoke actions for sustained state changes only. Amazon CloudWatch alarms will not invoke actions simply because they are in a particular state, the state must have changed and been maintained for a specified number of periods.

Related Topics

- `PutMetricAlarm` (Amazon CloudWatch API Reference)
- mon-put-metric-alarm Command (p. 118)
- Creating CloudWatch Alarms (p. 65)

For examples of setting up Amazon CloudWatch alarms that invoke an Auto Scaling policy and SNS topic, see Creating Amazon CloudWatch Alarms (p. 65).

Regions

Amazon cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, each data center facility is located in a specific geographical area, known as a Region. Regions are large and widely dispersed geographic locations.

Each Amazon Region is designed to be completely isolated from the other Amazon Regions. This achieves the greatest possible failure independence and stability, and it makes the locality of each Amazon resource unambiguous. Amazon CloudWatch does not aggregate data across Regions. Therefore, metrics are completely separate between Regions.

For a list of endpoints that represent each region, go to Regions and Endpoints in the Amazon Web Services General Reference.

Choosing A CloudWatch Interface

Topics

You can access Amazon CloudWatch using several different interfaces—you can sign on to the AWS Management Console, download and install the Command Line Interface (CLI), or create a query request with the Query API. If you prefer to use a specific programming language, several Software Development Kits (SDKs) exist that allow you to access Amazon CloudWatch programmatically.

AWS Management Console

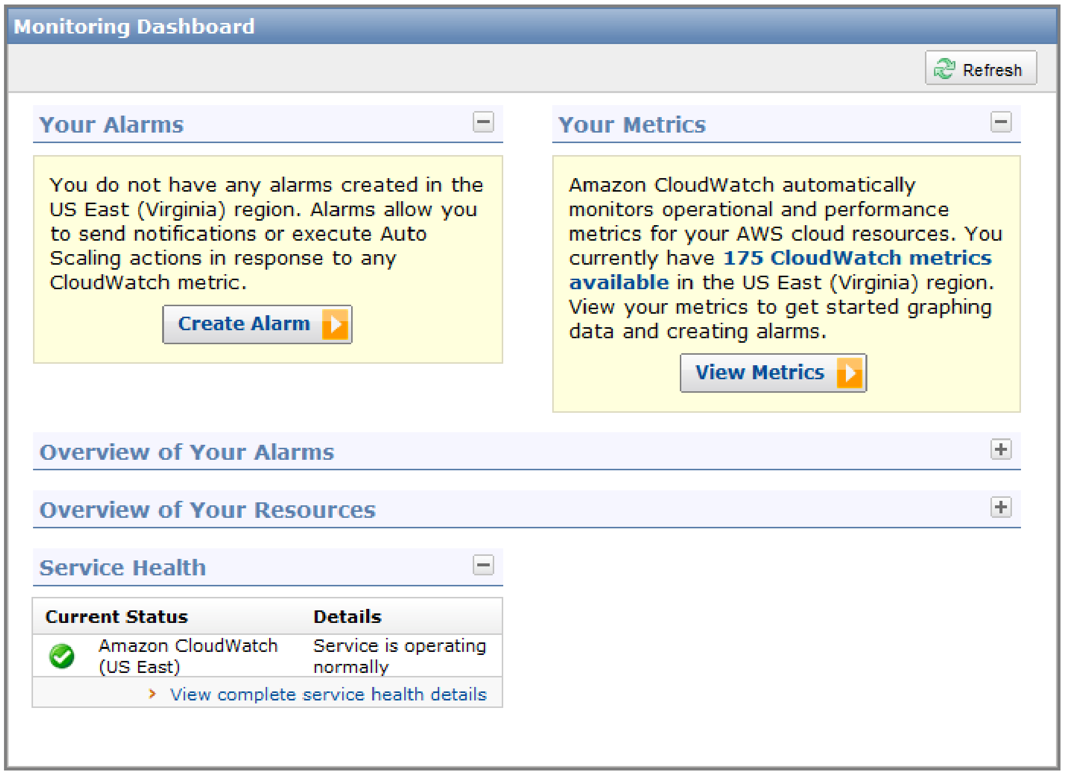
Topics

Use the Amazon CloudWatch console to view graphs of your metrics and create alarms based on those metrics. If you are registered with an AWS service that supports data collection, you could already have basic metric data available to you in Amazon CloudWatch.

Signing in to the Console

To sign in to the CloudWatch console

- Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
  The monitoring dashboard opens. Your dashboard might look something like the following:



You do not have any alarms created in the US East (Virginia) region. Alarms allow you to send notifications or execute Auto Scaling actions in response to any CloudWatch metric.

Amazon CloudWatch automatically monitors operational and performance metrics for your AWS cloud resources. You currently have **175 CloudWatch metrics available** in the US East (Virginia) region. View your metrics to get started graphing data and creating alarms.

Current Status		Details
✓	Amazon CloudWatch (US East)	Service is operating normally

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

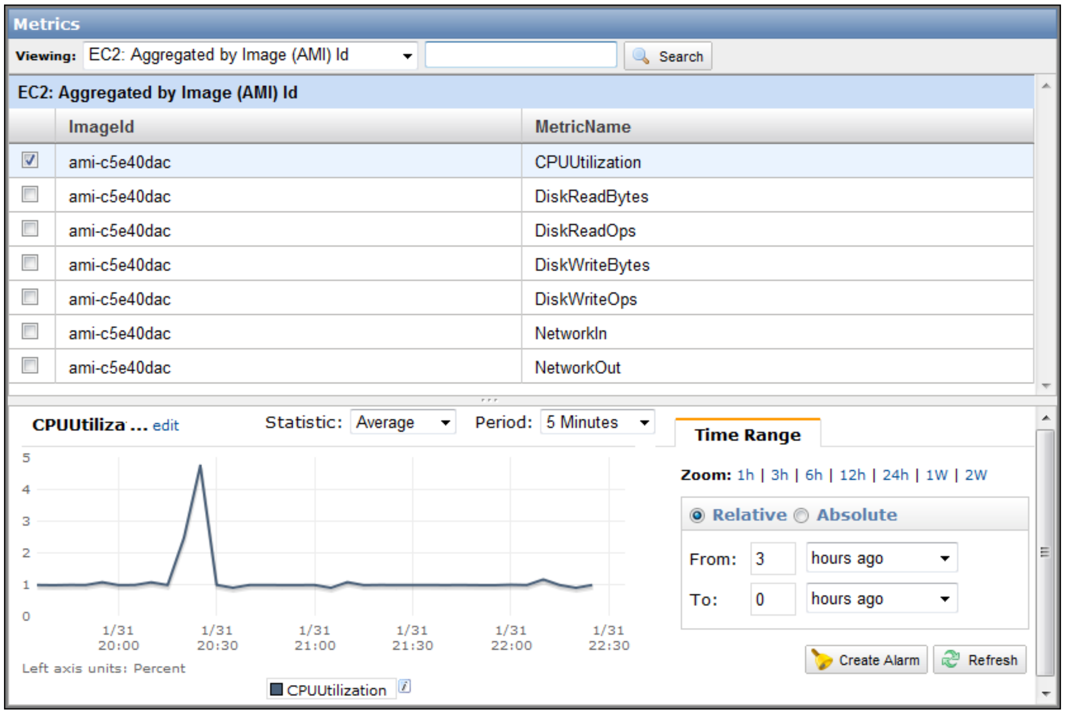
Viewing Your Metrics

You can view graphs of your metrics in the **Metrics** page.

To see graphs of your metrics

1. In the **Navigation** pane, click **Metrics**.
2. Scroll down to the metric that you want to graph.
3. Click the metric.

The following example shows the graph for the `CPUUtilization` metric, aggregated by EC2 Image ID.



Metrics

Viewing: EC2: Aggregated by Image (AMI) Id

EC2: Aggregated by Image (AMI) Id

	ImageId	MetricName
✓	ami-c5e40dac	CPUUtilization
	ami-c5e40dac	DiskReadBytes
	ami-c5e40dac	DiskReadOps
	ami-c5e40dac	DiskWriteBytes
	ami-c5e40dac	DiskWriteOps
	ami-c5e40dac	NetworkIn
	ami-c5e40dac	NetworkOut

CPUUtiliza... edit   Statistic: Average   Period: 5 Minutes

Time Range

Zoom: 1h | 3h | 6h | 12h | 24h | 1W | 2W

[✓ Relative] Absolute

From: 3 hours ago

To: 0 hours ago

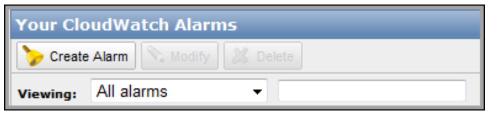
Left axis units: Percent

Creating Alarms

You can create an alarm in the **Your CloudWatch Alarms** page.

To create an alarm

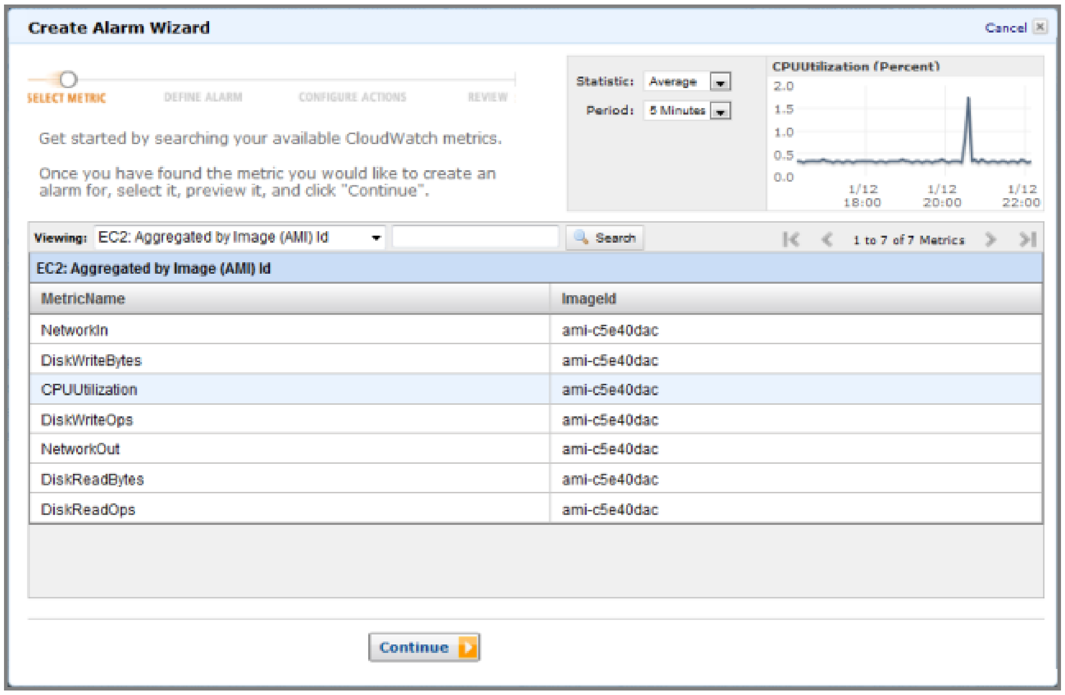
1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Alarms**.
   The **Your CloudWatch Alarms** page opens.



Your CloudWatch Alarms

Viewing: All alarms

3. Click **Create Alarm**.
   The **Create Alarm Wizard** page opens.



Get started by searching your available CloudWatch metrics.

Once you have found the metric you would like to create an alarm for, select it, preview it, and click "Continue".

Statistic: Average

Period: 5 Minutes

Viewing: EC2: Aggregated by Image (AMI) Id

EC2: Aggregated by Image (AMI) Id

MetricName	ImageId
NetworkIn	ami-c5e40dac
DiskWriteBytes	ami-c5e40dac
CPUUtilization	ami-c5e40dac
DiskWriteOps	ami-c5e40dac
NetworkOut	ami-c5e40dac
DiskReadBytes	ami-c5e40dac
DiskReadOps	ami-c5e40dac

The **Create Alarm Wizard** leads you through the steps to create a new alarm. You can choose specific metrics to trigger the alarm and specify thresholds for those metrics. You can then set your alarm to change state when a metric breaches a threshold that you have defined.

After creating an alarm, the alarm appears in the **Your CloudWatch Alarms** section of CloudWatch. For an example of how to create an alarm for a specific metric, see Creating CloudWatch Alarms (p. 65).

Command Line Tools

Topics

- Installing the Command Line Tool (p. 13)
- Command Line Tools Example (p. 18)

The following sections describe how to set up your environment for use with the Amazon CloudWatch command line tool.

Installing the Command Line Tool

This section describes how to set up the Amazon CloudWatch command line tool.

Process for Installing the Command Line Tool

Task 1: Download the Command Line Tool (p. 14)
Task 2: Set the JAVA_HOME Environment Variable (p. 14)
Task 3: Set the AWS_CLOUDWATCH_HOME Environment Variable (p. 15)
Task 4: Set the AWS_CREDENTIAL_FILE Environment Variable (p. 16)

Note

As a convention, command line text is prefixed with a generic **`PROMPT>`** command line prompt. The actual command line prompt on your computer is likely to be different. We also use **`$`** to indicate a Linux/UNIX–specific command and **`C:\>`** for a Windows–specific command. Although we don't provide explicit instructions, the tool also works correctly on Mac OS X (which resemble the Linux and UNIX commands). The example output resulting from the command is shown immediately thereafter without any prefix.

Task 1: Download the Command Line Tool

The command line tool is available as a ZIP file on the Amazon CloudWatch Developer Tools website. The tool is written in Java and includes shell scripts for both Windows and Linux/UNIX/Mac OSX. The ZIP file is self-contained; no installation is required. You just download it and unzip it.

Some additional setup is required before you can use the tool. These steps are discussed next.

Task 2: Set the JAVA_HOME Environment Variable

The Amazon CloudWatch command line tool reads an environment variable (`JAVA_HOME`) on your computer to locate the Java runtime. The command line tool requires Java version 5 or later to run. Either a JRE or JDK installation is acceptable.

To set the `JAVA_HOME` Environment Variable

1. If you do not have Java 1.5 or later installed, download and install Java. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, go to http://java.oracle.com/.
2. Set `JAVA_HOME` to the full path of the directory that contains a subdirectory named `bin` that in turn contains the Java executable. For example, if your Java executable is in the `/usr/jdk/bin` directory, set `JAVA_HOME` to `/usr/jdk`. If your Java executable is in `C:\jdk\bin`, set `JAVA_HOME` to `C:\jdk`.

   Note

   If you are using Cygwin, you must use Linux/UNIX paths (e.g., /usr/bin instead of C:\usr\bin) for `AWS_CLOUDWATCH_HOME` and `AWS_CREDENTIAL_FILE`. However, `JAVA_HOME` should have a Windows path. Additionally, the value cannot contain any spaces, even if the value is quoted or the spaces are escaped.

   The following Linux/UNIX example sets `JAVA_HOME` for a Java executable in the `/usr/local/jre/bin` directory.

   ```
   $ export JAVA_HOME=/usr/local/jre
   ```

   The following Windows example uses **set** and **setx** to set `JAVA_HOME` for a Java executable in the `C:\java\jdk1.6.0_6\bin` directory. The **set** command defines `JAVA_HOME` for the current session and **setx** makes the change permanent.

   ```
   C:\> set JAVA_HOME=C:\java\jdk1.6.0_6
   C:\> setx JAVA_HOME C:\java\jdk1.6.0_6
   ```

Note

Don't include the bin directory in JAVA_HOME; that's a common mistake some users make. The command line tool won't work if you do.

3. Add your Java directory to your path before other versions of Java.

   On Linux and UNIX, you can update your PATH as follows:

```
$ export PATH=$JAVA_HOME/bin:$PATH
```

   On Windows the syntax is slightly different:

```
C:\> set PATH=%JAVA_HOME%\bin;%PATH%
C:\> setx PATH %JAVA_HOME%\bin;%PATH%
```

Note

The **setx** command does not use the "=" sign.

4. Verify your JAVA_HOME setting with the command **$JAVA_HOME/bin/java -version**.

```
$ $JAVA_HOME/bin/java -version
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

   The syntax is different on Windows, but the output is similar.

```
C:\> %JAVA_HOME%\bin\java -version
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03)
Java HotSpot(TM) Client VM (build 1.5.0_09-b03, mixed mode, sharing)
```

Task 3: Set the AWS_CLOUDWATCH_HOME Environment Variable

The command line tool depends on an environment variable (AWS_CLOUDWATCH_HOME) to locate supporting libraries. You'll need to set this environment variable before you can use the tool.

To set the AWS_CLOUDWATCH_HOME Environment Variable

1. Set AWS_CLOUDWATCH_HOME to the path of the directory into which you unzipped the command line tool. This directory is named -w.x.y.z (w, x, y, and z are version/release numbers) and contains sub-directories named bin and lib.

   The following Linux/UNIX example sets AWS_CLOUDWATCH_HOME for a directory named -1.0.12.0 in the /usr/local directory.

```
$ export AWS_CLOUDWATCH_HOME=/usr/local/-1.0.12.0
```

   The following Windows example sets AWS_CLOUDWATCH_HOME for a directory named -1.0.12.0 in the C:\CLIs directory.

```
C:\> set AWS_CLOUDWATCH_HOME=C:\CLIs\-1.0.12.0
C:\> setx AWS_CLOUDWATCH_HOME C:\CLIs\-1.0.12.0
```

2. Add the tool's `bin` directory to your system `PATH`. The rest of this guide assumes that you've done this.

   On Linux and UNIX, you can update your `PATH` as follows:

   ```
   $ export PATH=$PATH:$AWS_CLOUDWATCH_HOME/bin
   ```

   On Windows the syntax is slightly different:

   ```
   C:\> set PATH=%PATH%;%AWS_CLOUDWATCH_HOME%\bin
   C:\> setx PATH %PATH%;%AWS_CLOUDWATCH_HOME%\bin
   ```

Task 4: Set the AWS_CREDENTIAL_FILE Environment Variable

You must also provide your AWS credentials to the command line tool. The command line tool reads your credentials from a credential file that you create on your local system.

You can either specify your credentials with the `--aws-credential-file` parameter every time you issue a command or you can create an environment variable that points to the credential file on your local system. If the environment variable is properly configured, you can omit the `--aws-credential-file` parameter when you issue a command. The following procedure describes how to create a credential file and a corresponding `AWS_CREDENTIAL_FILE` environment variable.

To set up security credentials for your command line tool

1. Log in to the AWS security credentials web site.
2. Retrieve an access key and its corresponding secret key.
   a. Scroll down to the **Access Credentials** section and select the **Access Keys** tab.
   b. Locate an active Access Key in the **Your Access Keys** list.
   c. To display the Secret Access Key, click **Show** in the **Secret Access Key** column.
   d. Write down the keys or save them.
   e. If no Access Keys appear in the list, click **Create a New Access Key** and follow the on-screen prompts.
3. Add your access key ID and secret access key to the file named `credential-file-path.template`:
   a. Open the file `credential-file-path.template` included in your command line tools archive.
   b. Copy and paste your access key ID and secret access key into the file.
   c. Rename the file and save it to a convenient location on your computer.
   d. If you are using Linux, set the file permissions as follows:

   ```
   $ chmod 600 credential-file-name
   ```

4. Set the AWS_CREDENTIAL_FILE environment variable to the fully qualified path of the file you just created.

   The following Linux/UNIX example sets AWS_CREDENTIAL_FILE for myCredentialFile in the /usr/local directory.

   ```
   $ export AWS_CREDENTIAL_FILE=/usr/local/myCredentialFile
   ```

   The following Windows example sets AWS_CREDENTIAL_FILE for myCredentialFile.txt in the C:\aws directory.

   ```
   C:\> set AWS_CREDENTIAL_FILE=C:\aws\myCredentialFile.txt
   C:\> setx AWS_CREDENTIAL_FILE C:\aws\myCredentialFile.txt
   ```

Task 5: Set the Region

By default, the Amazon CloudWatch tools use the US East (Northern Virginia) Region (us-east-1) with the monitoring.us-east-1.amazonaws.com service endpoint URL. If your instances are in a different region, you must specify the region where your instances reside. For example, if your instances are in Europe, you must specify the EU (Ireland) Region by using the --region eu-west-1 parameter or by setting the AWS_CLOUDWATCH_URL environment variable.

This section describes how to specify a different Region by changing the service endpoint URL.

To specify a different Region

1. To view available Regions go to Regions and Endpoints in the *Amazon Web Services General Reference*.
2. If you want to change the service endpoint, set the AWS_CLOUDWATCH_URL environment variable.
   - The following Linux/UNIX example sets AWS_CLOUDWATCH_URL to the EU (Ireland) Region.

   ```
   $ export AWS_CLOUDWATCH_URL=https://monitoring.eu-west-1.amazonaws.com
   ```

   - The following Windows example sets AWS_CLOUDWATCH_URL to the EU (Ireland) Region.

   ```
   C:\> set AWS_CLOUDWATCH_URL=https://monitoring.eu-west-1.amazonaws.com
   C:\> setx AWS_CLOUDWATCH_URL https://monitoring.eu-west-1.amazonaws.com
   ```

You're ready to start using Amazon CloudWatch.

Command Line Tools Example

This section shows some examples of command line tools usage.

Note

This section uses command line tools for Amazon CloudWatch and Amazon EC2. For more information about the Amazon EC2 command line tools, go to Getting Started with the Command Line Tools in the *Amazon Elastic Compute Cloud User Guide*.

- Use the EC2 `ec2-run-instances` command as in the following example.

```
PROMPT>ec2-run-instances ami-60a54009 -n 3 --availability-zone us-east-1a
```

Note

If you wanted to automatically monitor all of the EC2 Instances spun up by the preceding command, you would add `--monitoring`. This lets you skip step 2 in this procedure.

The command returns a unique identifier for each launched instance. You use the instance ID to manipulate the instance. This includes viewing the status of the instance, terminating the instance, and so on. Launching the instance takes a few minutes.

```
RESERVATION r-237fed4a 853279305796 default
INSTANCE i-d9add0b0 ami-60a54009 pending 0 m1.small 2009-05-14T12:38:24+0000
 us-east-1a aki-a71cf9ce ari-a51cf9cc monitoring-disabled
```

- Use the `mon-list-metrics` command to get a list of what metrics are being stored for your AWS account. For example:

```
PROMPT>mon-list-metrics
```

The `mon-list-metrics` outputs a table containing the Metric name, Namespace, and Dimension associated with each metric. For example:

```
CPUUtilization       AWS/EC2  {InstanceId=i-c385b3aa}
CPUUtilization       AWS/EC2  {ImageId=ami-11ca2d78}
CPUUtilization       AWS/EC2  {InstanceType=m1.small}
CPUUtilization       AWS/EC2
DiskReadBytes        AWS/EC2  {ImageId=ami-11ca2d78}
DiskReadBytes        AWS/EC2  {InstanceType=m1.small}
DiskReadBytes        AWS/EC2
DiskReadBytes        AWS/EC2  {InstanceId=i-c385b3aa}
DiskReadOps          AWS/EC2  {InstanceId=i-c385b3aa}
DiskReadOps          AWS/EC2  {InstanceType=m1.small}
DiskReadOps          AWS/EC2
DiskReadOps          AWS/EC2  {ImageId=ami-11ca2d78}
DiskWriteBytes       AWS/EC2  {InstanceId=i-c385b3aa}
DiskWriteBytes       AWS/EC2  {InstanceType=m1.small}
DiskWriteBytes       AWS/EC2  {ImageId=ami-11ca2d78}
DiskWriteBytes       AWS/EC2
DiskWriteOps         AWS/EC2  {InstanceId=i-c385b3aa}
DiskWriteOps         AWS/EC2  {ImageId=ami-11ca2d78}
DiskWriteOps         AWS/EC2  {InstanceType=m1.small}
DiskWriteOps          AWS/EC2
NetworkIn             AWS/EC2
NetworkIn             AWS/EC2  {InstanceId=i-c385b3aa}
NetworkIn             AWS/EC2  {ImageId=ami-11ca2d78}
NetworkIn             AWS/EC2  {InstanceType=m1.small}
NetworkOut            AWS/EC2  {InstanceType=m1.small}
NetworkOut            AWS/EC2
NetworkOut            AWS/EC2  {ImageId=ami-11ca2d78}
NetworkOut            AWS/EC2  {InstanceId=i-c385b3aa}
```

- Use the EC2 `ec2-monitor-instances` command as in the following example.

```
PROMPT> ec2-monitor-instances i-43a4412a
```

`ec2-monitor-instances` returns a table that contains the selected instance IDs and the current monitoring state.

```
i-43a4412a monitoring-pending
```

- Use the Amazon CloudWatch `mon-get-stats` command as in the following example.

```
PROMPT> mon-get-stats CPUUtilization --start-time 2009-05-15T00:00:00 --end-
time 2009-05-16T00:00:00 --period 60 --statistics "Average"
--namespace "AWS/EC2" --dimensions "ImageId=ami-60a54009"
```

Amazon CloudWatch returns a response similar to the following (the data has been truncated for brevity):

```
2009-05-15  22:42:00    0.38   Percent
2009-05-15  22:48:00    0.39   Percent
2009-05-15  22:54:00    0.38   Percent
```

Note

Amazon CloudWatch returns the data for this function in the following order: date, time, sample, CPUUtilization, and unit.

Query API

Query requests are HTTP or HTTPS requests that use the HTTP verb GET or POST and a Query parameter named Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs.

Amazon CloudWatch Endpoints

For information about this product's regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

Query Parameters

Each Query request must include some common parameters to handle authentication and selection of an action. For more information, see Common Query Parameters in the *Amazon CloudWatch API Reference*.

Note

Some API operations take lists of parameters. These lists are specified using the following notation: `param.member.n`. Values of n are integers starting from 1. All lists of parameters must follow this notation, including lists that contain only one parameter. For example, a Query parameter list looks like this:

```
&attribute.member.1=this
&attribute.member.2=that
```

The RequestId

In every response from Amazon Web Services (AWS), you will find `ResponseMetadata`, which contains a string element called `RequestId`. This is simply a unique identifier that AWS assigns to provide tracking information. Although `RequestId` is included as part of every response, it will not be listed on the individual API documentation pages to improve readability of the API documentation and to reduce redundancy.

Query API Authentication

You can send Query requests over either HTTP or HTTPS. Regardless of which protocol you use, you must include a signature in every Query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 2*.

To create the signature

1. Create the canonicalized query string that you will need later in this procedure:

   a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when `Content-Type` is `application/x-www-form-urlencoded`).

   b. URL-encode the parameter name and values according to the following rules:

      - Do not URL-encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
      - Percent-encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
      - Percent-encode extended UTF-8 characters in the form %XY%ZA and so on.
      - Percent-encode the space character as %20 (and not +, as common encoding schemes do).

      Note

      Currently, all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

   c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII code 61), even if the parameter value is empty.

   d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the `"\n"` represents an ASCII newline).

   ```
   StringToSign = HTTPVerb + "\n" +
   ValueOfHostHeaderInLowercase + "\n" +
   HTTPRequestURI + "\n" +
   CanonicalizedQueryString <from the preceding step>
   ```

   The `HTTPRequestURI` component is the HTTP absolute path component of the URI up to but not including the query string. If the `HTTPRequestURI` is empty, use a forward slash (`/`).
3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key (p. 141) as the key, and SHA256 or SHA1 as the hash algorithm.
   For more information, go to http://www.ietf.org/rfc/rfc2104.txt.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the *`Signature`* request parameter.

Important

The final signature you send in the request must be URL-encoded as specified in RFC 3986 (for more information, go to http://www.ietf.org/rfc/rfc3986.txt). If your toolkit URL-encodes your final request, then it handles the required URL-encoding of the signature. If your toolkit doesn't URL-encode the final request, then make sure to URL-encode the signature before you include it in the request. Most importantly, make sure the signature is URL-encoded *only once*. A common mistake is to URL-encode it manually during signature formation, and then again when the toolkit URL-encodes the entire request.

Query API

Example ListMetrics API Request

This example uses the Amazon CloudWatch ListMetrics.

```
http://monitoring.amazonaws.com/?SignatureVersion=2
&Action=ListMetrics
&Version=2010-08-01
&AWSAccessKeyId=<Your AWS Access Key Id>
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-11-17T05%3A13%3A00.000Z
```

The following is the string to sign.

```
GET\n
monitoring.amazonaws.com\n
/\n
AWSAccessKeyId=<Your AWS Access Key Id>
&Action=ListMetrics
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2010-11-17T05%3A13%3A00.000Z
&Version=2010-08-01
```

The following is the signed request.

```
http://monitoring.amazonaws.com/?Action=ListMetrics
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-11-17T05%3A13%3A00.000Z
&Signature=<URLEncode(Base64Encode(Signature))>
&Version=2010-08-01
&AWSAccessKeyId=<Your AWS Access Key Id>
```

Available Libraries

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of the command-line tools and Query. These libraries provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. Libraries and resources are available for the following languages and platforms:

- Android
- iOS
- Java
- PHP
- Python
- Ruby
- Windows and .NET

For libraries and sample code in all languages, go to Sample Code & Libraries.

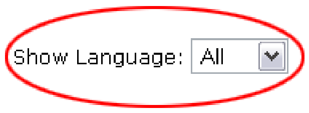
Using Amazon CloudWatch

Topics

This section describes how to use Amazon CloudWatch. You can access Amazon CloudWatch by signing on to the AWS Management Console, downloading and installing the Command Line Interface (CLI), or creating a query request with the Query API.

This section expands on the basic concepts presented in the preceding section (see Introduction to Amazon CloudWatch (p. 2)), and includes procedures for using Amazon CloudWatch. This section also shows you how to view metrics that other AWS products provide to Amazon CloudWatch and how to publish custom metrics with Amazon CloudWatch.

The procedures in this section include instructions for the AWS Management Console, the command line tools (i.e., the API tools), and the Query API. In the HTML version of this document, you can show instructions for a single interface. There is an interface selection menu in the upper-right corner of each web page. Select your interface of choice to hide all others, or select All to show the instructions in all available interfaces.



Viewing Your AWS Metrics With Amazon CloudWatch

Topics

- Listing Available Metrics (p. 25)
- Getting Statistics for a Metric (p. 27)
- Amazon CloudWatch Metrics, Namespaces, and Dimensions Reference (p. 36)

Amazon CloudWatch offers basic monitoring for several AWS products. Basic monitoring means that a service sends data points to Amazon CloudWatch every five minutes. Amazon CloudWatch offers detailed monitoring for Amazon EC2 and Auto Scaling. Detailed monitoring means that a service sends data points to Amazon CloudWatch every minute. Amazon CloudWatch currently provides basic monitoring for the following services:

- Amazon DynamoDB (DynamoDB) begins sending basic statistics automatically when you begin using it. For more information, see Amazon DynamoDB Dimensions and Metrics (p. 38).
- Amazon Elastic Compute Cloud (Amazon EC2) begins sending basic statistics automatically when you launch any instance (including instances of paid AMIs). You can opt for detailed monitoring. For more information, see Amazon Elastic Compute Cloud Dimensions and Metrics (p. 44).
- Amazon Elastic Block Store (Amazon EBS) begins sending basic statistics automatically when you mount the volume. For more information, see Amazon EBS Dimensions and Metrics (p. 43).
- Amazon Elastic Map Reduce (Amazon EMR) begins sending basic statistics automatically when you create a job flow. For more information, see Amazon Elastic MapReduce Dimensions and Metrics (p. 48).
- Amazon Relational Database Service (Amazon RDS) begins sending RDS-specific statistics when you begin using RDS. For more information, see Amazon RDS Dimensions and Metrics (p. 52).
- Elastic Load Balancing begins sending statistics when you begin using it. For more information, see Elastic Load Balancing Dimensions and Metrics (p. 62).
- Auto Scaling begins sending statistics when you begin using it. You can opt for detailed monitoring. For more information, see Auto Scaling Dimensions and Metrics (p. 59).

Note

If you are registered with an AWS product that supports both basic and detailed data collection, and want to access detailed statistics, you must enable detailed metric collection for that service. For more information, see Activating Detailed Monitoring for Amazon EC2 (p. 47).

Listing Available Metrics

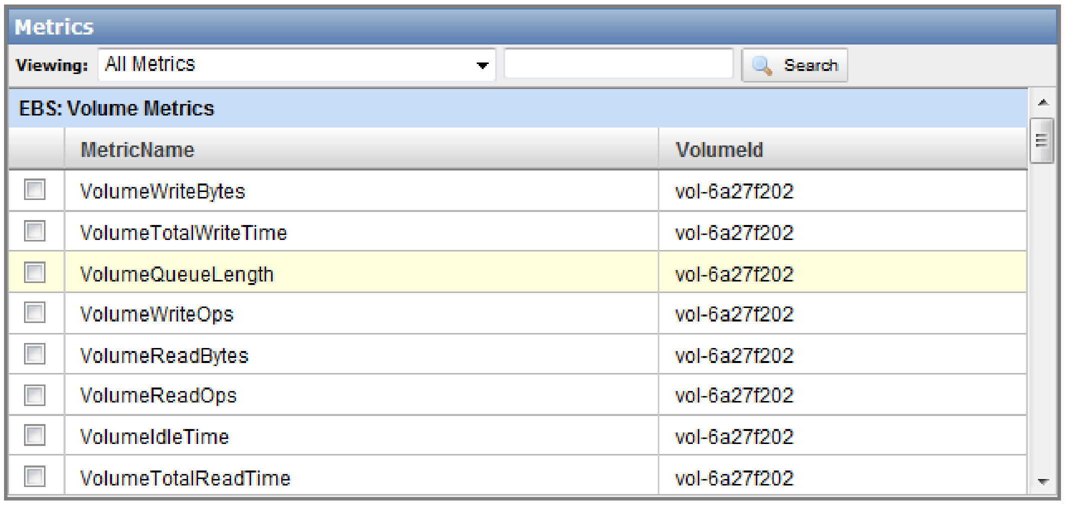
You can use the AWS Management Console, the mon-list-metrics command, or the `ListMetrics` API to determine which metrics are available.

AWS Management Console

To display available metrics across multiple instances

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Metrics**.

   The **Metrics** page opens.



EBS: Volume Metrics

	MetricName	VolumeId
☐	VolumeWriteBytes	vol-6a27f202
☐	VolumeTotalWriteTime	vol-6a27f202
☐	VolumeQueueLength	vol-6a27f202
☐	VolumeWriteOps	vol-6a27f202
☐	VolumeReadBytes	vol-6a27f202
☐	VolumeReadOps	vol-6a27f202
☐	VolumeIdleTime	vol-6a27f202
☐	VolumeTotalReadTime	vol-6a27f202

3. Scroll down to view the full list of metrics.

Command Line Tools

To list available metrics across multiple instances

- Enter the `mon-list-metrics` command. Add the `--headers` parameter to display column headings.

```
Prompt>mon-list-metrics --headers
```

Amazon CloudWatch returns the following (partial listing):

```
Metric Name         Namespace  Dimensions
CPUUtilization      AWS/EC2    {InstanceId=i-5431413d}
CPUUtilization      AWS/EC2    {InstanceId=i-d43242bd}
CPUUtilization      AWS/EC2    {InstanceId=i-1d3d4d74}
CPUUtilization      AWS/EC2    {InstanceId=i-78314111}
CPUUtilization      AWS/EC2    {InstanceId=i-d3c8baba}
CPUUtilization      AWS/EC2    {InstanceId=i-0d334364}
CPUUtilization      AWS/EC2    {InstanceId=i-6732420e}
CPUUtilization      AWS/EC2    {InstanceId=i-d93141b0}
CPUUtilization      AWS/EC2    {InstanceId=i-e03d4d89}
CPUUtilization      AWS/EC2    {InstanceId=i-c93d4da0}
CPUUtilization      AWS/EC2    {InstanceId=i-e0304089}
CPUUtilization      AWS/EC2    {InstanceId=i-e1304088}
CPUUtilization      AWS/EC2    {InstanceId=i-69334300}
```

Query API

To determine available metrics across multiple instances

- Call ListMetrics to generate a list of all of your valid metrics.

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

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

Getting Statistics for a Metric

Topics

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

Note

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

Getting Statistics Aggregated Across All Instances

This scenario shows you how to use either the AWS Management Console, the `GetMetricStatistics` API, or the `mon-get-stats` command to get the average CPU usage for all EC2 instances for the account. Because no dimension is specified, Amazon CloudWatch returns statistics for all dimensions in the `AWS/EC2` namespace.

Important

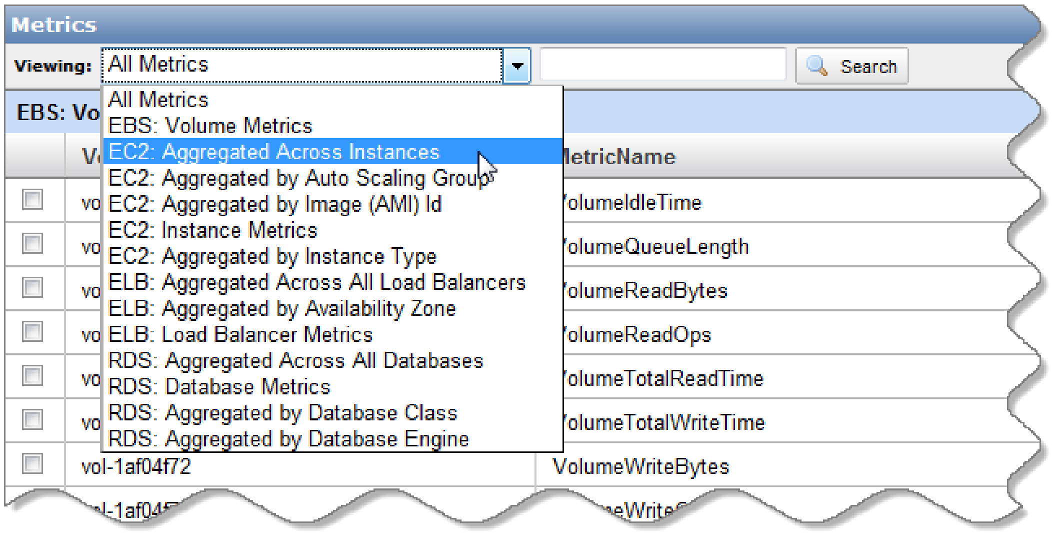
This technique for retrieving all dimensions across an AWS namespace does not work for custom namespaces that you publish to Amazon CloudWatch. With custom namespaces, you must specify the complete set of dimensions that are associated with any given data point to retrieve statistics that include the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

AWS Management Console

To display average CPU utilization for all your EC2 instances

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Metrics**.

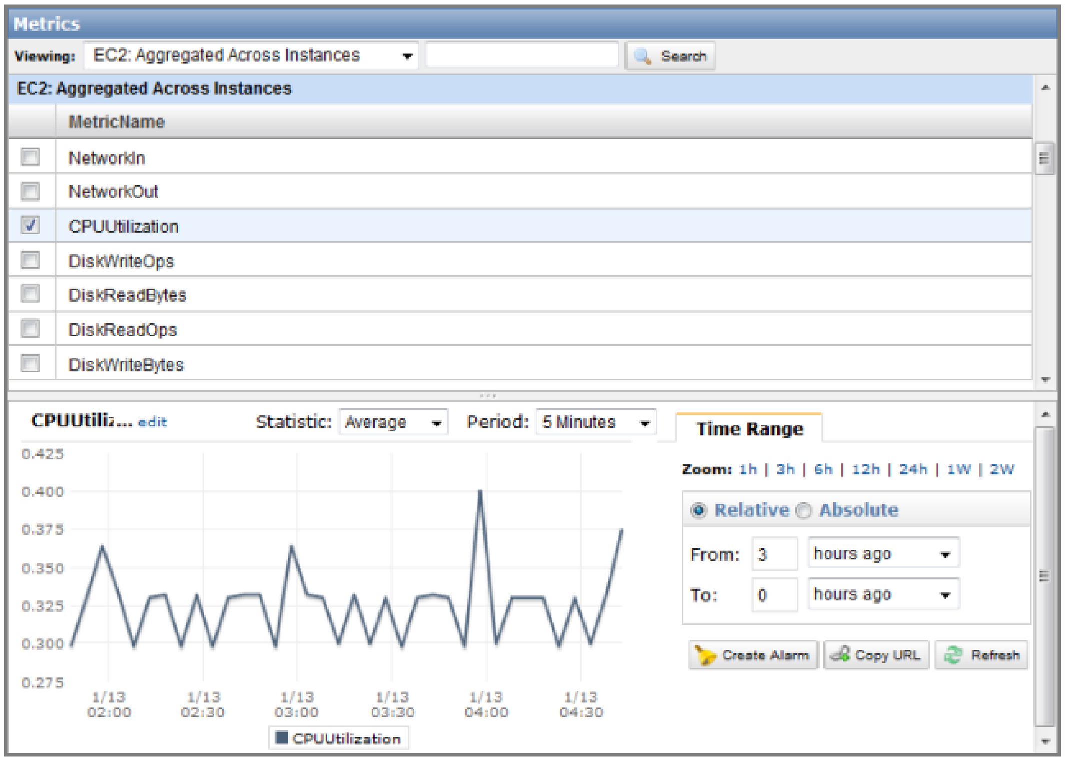
   The **Metrics** page opens.


Metrics
Viewing: All Metrics
Search
All Metrics
EBS: Volume Metrics
EC2: Aggregated Across Instances
EC2: Aggregated by Auto Scaling Group
EC2: Aggregated by Image (AMI) Id
EC2: Instance Metrics
EC2: Aggregated by Instance Type
ELB: Aggregated Across All Load Balancers
ELB: Aggregated by Availability Zone
ELB: Load Balancer Metrics
RDS: Aggregated Across All Databases
RDS: Database Metrics
RDS: Aggregated by Database Class
RDS: Aggregated by Database Engine
vol-1af04f72
VolumeWriteBytes

3. Select **EC2: Aggregated Across Instances** from the **Viewing** drop-down list.

   The metrics available across all instances appear.
4. Select the row that contains **CPUUtilization**.

   A graph showing `CPUUtilization` for all EC2 instances appears.


Viewing: EC2: Aggregated Across Instances

EC2: Aggregated Across Instances

	MetricName
☐	NetworkIn
☐	NetworkOut
✓	CPUUtilization
☐	DiskWriteOps
☐	DiskReadBytes
☐	DiskReadOps
☐	DiskWriteBytes

Statistic: Average
Period: 5 Minutes
From: 3 hours ago
To: 0 hours ago

Command Line Tools

To get average CPU utilization across all Amazon EC2 instances

- Enter the `mon-get-stats` command with the following parameters:

```
Prompt>mon-get-stats CPUUtilization --start-time 2011-01-10T23:18:00 --end-
time 2011-01-12T23:18:00 --period 360 --namespace "AWS/EC2" --statistics
"Average,SampleCount" --headers
```

Amazon CloudWatch returns the following (partial listing):

```
Time                 SampleCount  Average  Unit
        2011-01-10 14:18:00  60.0         0.1138   Percent
        2011-01-10 15:18:00  60.0         0.1078   Percent
        2011-01-10 16:18:00  60.0         0.3322   Percent
        2011-01-10 17:18:00  60.0         0.1397   Percent
        2011-01-10 18:18:00  60.0         0.1143   Percent
        2011-01-10 19:18:00  60.0         0.1082   Percent
        2011-01-10 20:18:00  36.0         0.1367   Percent
```

Query API

To get average CPU utilization for all your EC2 instances

- Call `GetMetricStatistics` with the following parameters:

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

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

Getting Statistics Aggregated by Auto Scaling Group

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

Note

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

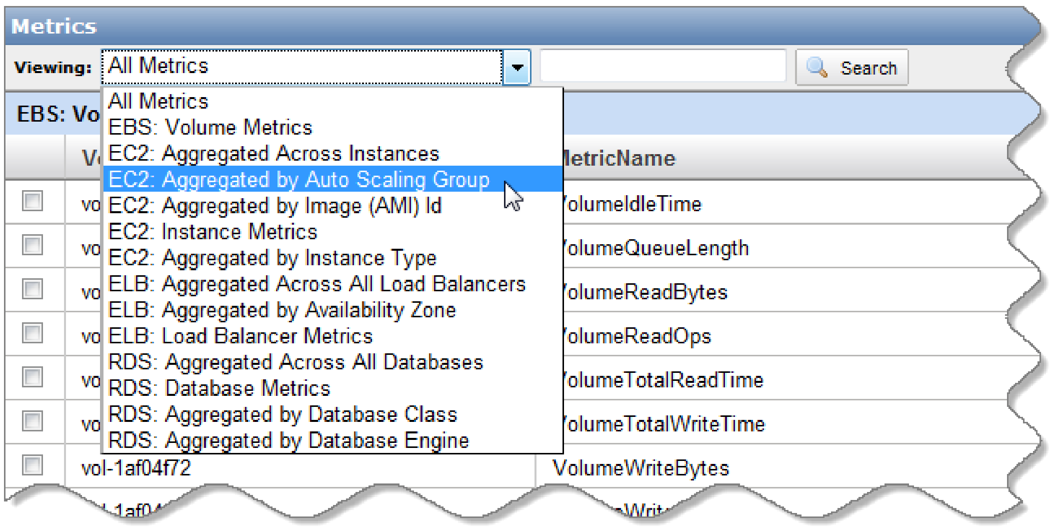
We assume for this example that an EC2 application is running and has an Auto Scaling group named `test-group-1`.

AWS Management Console

To display total DiskWriteBytes for an Auto-Scaled EC2 application

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Metrics**.

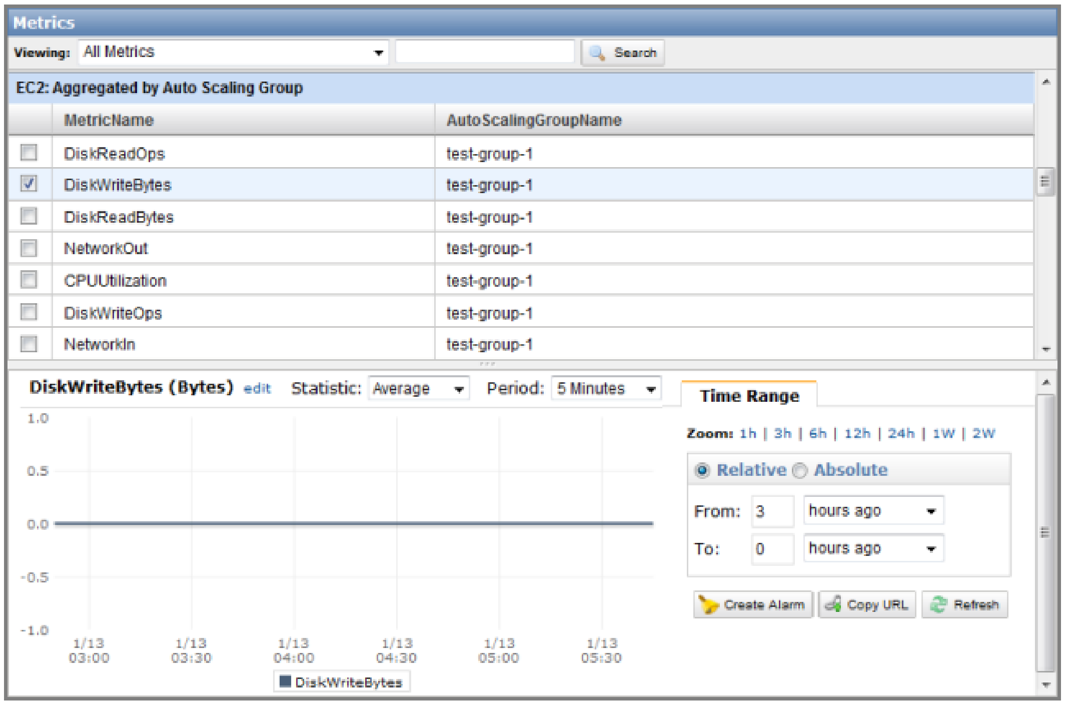
   The **Metrics** page opens.


Metrics
Viewing:
All Metrics
Search
EBS: Vo
MetricName
All Metrics
EBS: Volume Metrics
EC2: Aggregated Across Instances
EC2: Aggregated by Auto Scaling Group
EC2: Aggregated by Image (AMI) Id
EC2: Instance Metrics
EC2: Aggregated by Instance Type
ELB: Aggregated Across All Load Balancers
ELB: Aggregated by Availability Zone
ELB: Load Balancer Metrics
RDS: Aggregated Across All Databases
RDS: Database Metrics
RDS: Aggregated by Database Class
RDS: Aggregated by Database Engine
VolumeIdleTime
VolumeQueueLength
VolumeReadBytes
VolumeReadOps
VolumeTotalReadTime
VolumeTotalWriteTime
vol-1af04f72
VolumeWriteBytes

3. Select **EC2: Aggregated by Auto Scaling Group** from the **Viewing** drop-down list.

   The metrics available for Auto Scaling groups appear in the **Metrics** pane.
4. Select the row that contains **DiskWriteBytes**.

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



Command Line Tools

To get total DiskWriteBytes for an Auto-Scaled EC2 Application

- Enter the `mon-get-stats` command with the following parameters.

```
Prompt>mon-get-stats DiskWriteBytes --start-time 2011-01-10T23:18:00 --end-
time 2011-01-11T23:18:00 --period 60 --namespace AWS/EC2 --statistics
"Sum,SampleCount" --dimensions "AutoScalingGroupName=test-group-1" --headers
```

Amazon CloudWatch returns the following (partial listing):

```
Time                     SampleCount  Sum        Unit
        2011-01-10 15:52:00  1.0          196608.0   Bytes
        2011-01-10 15:53:00  1.0          180224.0   Bytes
        2011-01-10 15:54:00  1.0          200704.0   Bytes
        2011-01-10 15:55:00  1.0          200704.0   Bytes
        2011-01-10 15:56:00  1.0          200704.0   Bytes
        2011-01-10 15:57:00  1.0          180224.0   Bytes
        2011-01-10 15:58:00  1.0          196608.0   Bytes
        2011-01-10 15:59:00  1.0          372736.0   Bytes
        2011-01-10 16:00:00  1.0          258048.0   Bytes
```

Query API

To get total DiskWriteBytes for an Auto-Scaled EC2 Application

- Call `GetMetricStatistics` with the following parameters:

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

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

Getting Statistics Aggregated by Image (AMI) ID

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

Note

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

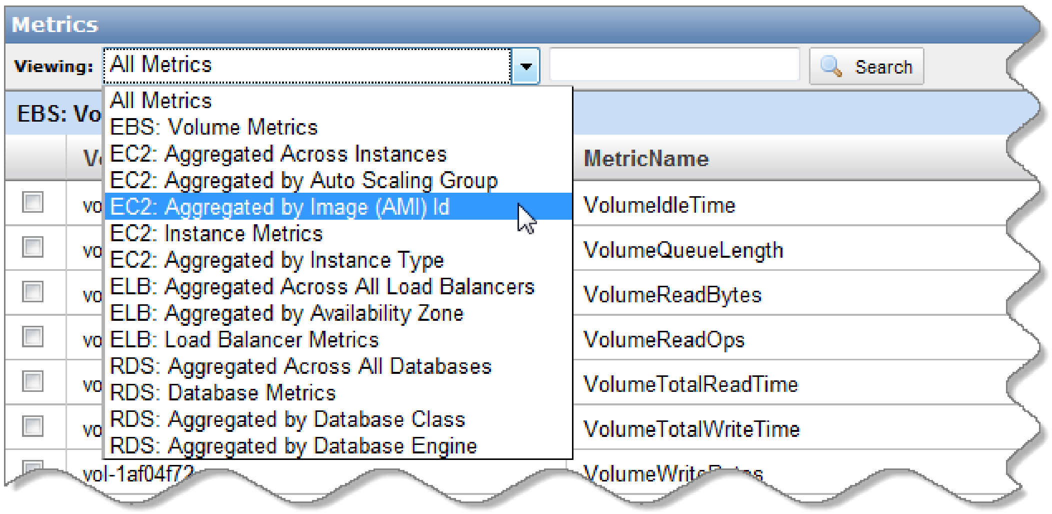
In this scenario the EC2 instances are running an image ID of `ami-c5e40dac`.

AWS Management Console

To display the average CPU utilization for an image ID

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Metrics**.

   The **Metrics** page opens.

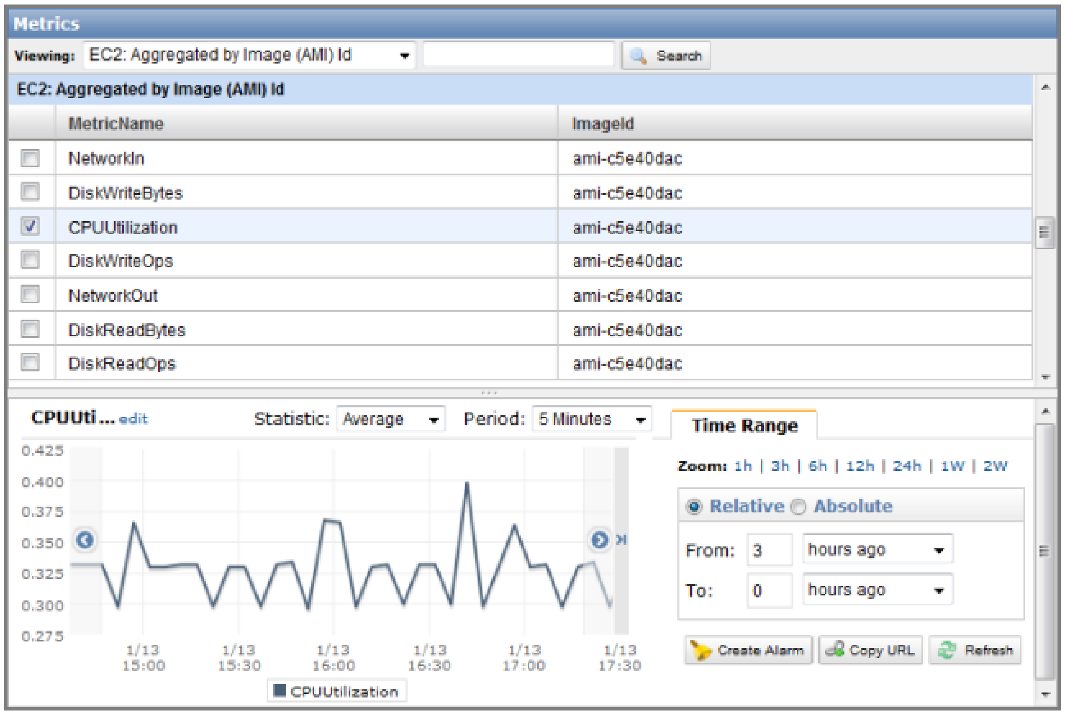

Metrics
Viewing:
All Metrics
Search
EBS: Vo
V
MetricName
All Metrics
EBS: Volume Metrics
EC2: Aggregated Across Instances
EC2: Aggregated by Auto Scaling Group
EC2: Aggregated by Image (AMI) Id
EC2: Instance Metrics
EC2: Aggregated by Instance Type
ELB: Aggregated Across All Load Balancers
ELB: Aggregated by Availability Zone
ELB: Load Balancer Metrics
RDS: Aggregated Across All Databases
RDS: Database Metrics
RDS: Aggregated by Database Class
RDS: Aggregated by Database Engine
VolumeIdleTime
VolumeQueueLength
VolumeReadBytes
VolumeReadOps
VolumeTotalReadTime
VolumeTotalWriteTime
vol-1af04f72

3. Select **EC2: Aggregated by Image (AMI) Id** from the **Viewing** drop-down list.

The metrics available for image IDs appear in the **Metrics** pane.

4. Select a row that contains **CPUUtilization** and an image ID.

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



	MetricName	ImageId
☐	NetworkIn	ami-c5e40dac
☐	DiskWriteBytes	ami-c5e40dac
✓	CPUUtilization	ami-c5e40dac
☐	DiskWriteOps	ami-c5e40dac
☐	NetworkOut	ami-c5e40dac
☐	DiskReadBytes	ami-c5e40dac
☐	DiskReadOps	ami-c5e40dac

Command Line Tools

To get the average CPU utilization for an image ID

- Enter the `mon-get-stats` command as in the following example.

```
Prompt>mon-get-stats CPUUtilization --start-time 2011-01-10T00:00:00 --end-
time 2011-01-11T00:00:00 --period 60 --statistics "Average" --namespace
"AWS/EC2" --dimensions "ImageId=ami-c5e40dac" --headers
```

Amazon CloudWatch returns the following:

```
Time                      Average  Unit
        2011-01-10 22:42:00   0.38      Percent
        2011-01-10 22:48:00   0.39      Percent
        2011-01-10 22:54:00   0.38      Percent
        2011-01-10 23:00:00   0.38      Percent
        2011-01-10 23:06:00   0.38      Percent
        2011-01-10 23:12:00   0.38      Percent
```

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

Query API

To get the average CPU utilization for an image ID

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

Getting Statistics for a Specific EC2 Instance

This scenario walks you through how to use the AWS Management Console, the `mon-get-stats` command, or the `GetMetricStatistics` API to determine the maximum CPU utilization of a specific EC2 instance.

Note

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

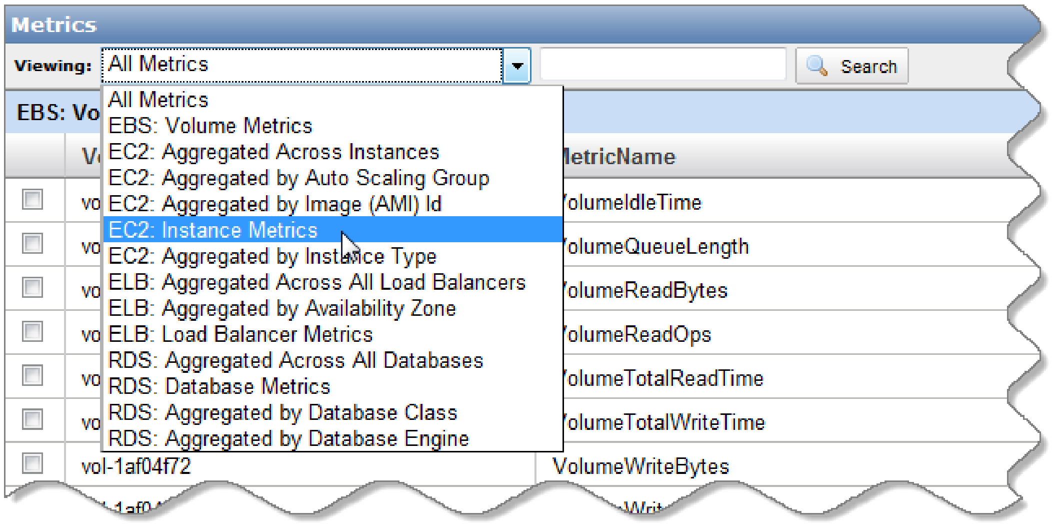
For this example, we assume that you have an EC2 instance ID. You can get an active EC2 instance ID through the AWS Management Console or with the EC2 `ec2-describe-instances` CLI command.

AWS Management Console

To display the average CPU utilization for a specific instance

1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Metrics** pane.

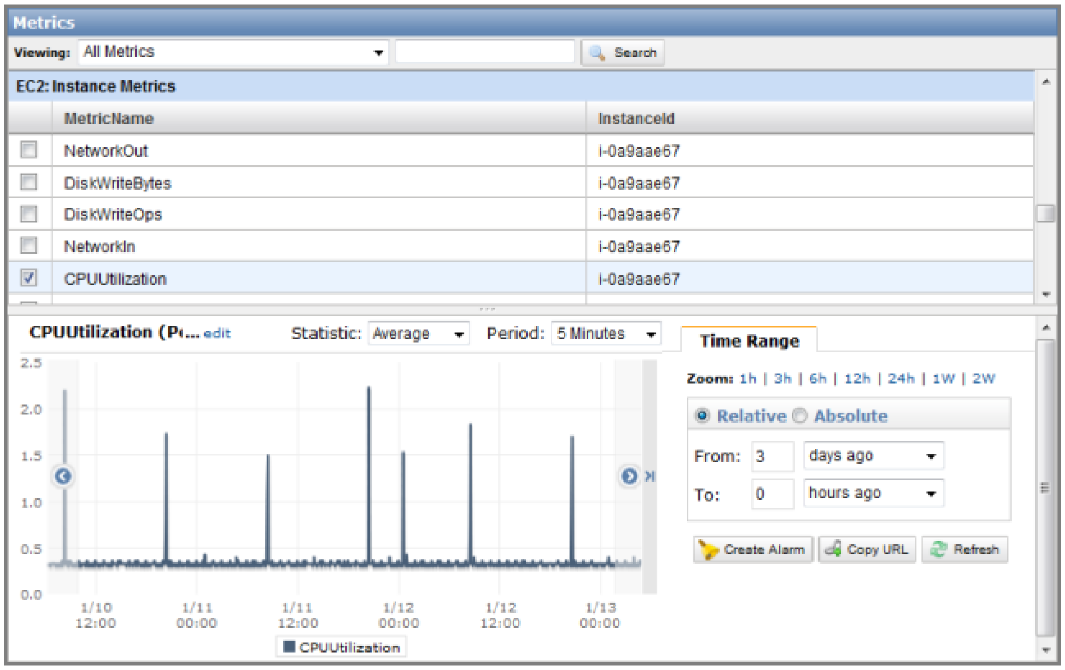
   The **Metrics** page opens.


Metrics
Viewing: All Metrics
Search
All Metrics
EBS: Volume Metrics
EC2: Aggregated Across Instances
EC2: Aggregated by Auto Scaling Group
EC2: Aggregated by Image (AMI) Id
EC2: Instance Metrics
EC2: Aggregated by Instance Type
ELB: Aggregated Across All Load Balancers
ELB: Aggregated by Availability Zone
ELB: Load Balancer Metrics
RDS: Aggregated Across All Databases
RDS: Database Metrics
RDS: Aggregated by Database Class
RDS: Aggregated by Database Engine
vol-1af04f72
VolumeWriteBytes

3. Select **EC2: Instance Metrics** from the **Viewing** drop-down box.

   The metrics available for individual instances appear in the **Metrics** pane.
4. Select a row that contains **CPUUtilization** for a specific InstanceId.

   A graph showing average `CPUUtilization` for a single instance appears in the **Metrics** pane.



Viewing: All Metrics

EC2: Instance Metrics

	MetricName	InstanceId
☐	NetworkOut	i-0a9aae67
☐	DiskWriteBytes	i-0a9aae67
☐	DiskWriteOps	i-0a9aae67
☐	NetworkIn	i-0a9aae67
☑	CPUUtilization	i-0a9aae67

Statistic: Average Period: 5 Minutes

Relative [✓] Absolute
From: 3 days ago
To: 0 hours ago

Command Line Tools

To get the CPU utilization per EC2 instance

- Enter the `mon-get-stats` command with the following parameters

```
Prompt>mon-get-stats CPUUtilization --start-time 2011-01-09T23:18:00 --end-
time 2011-01-12T23:18:00 --period 360 --namespace "AWS/EC2" --statistics
"Maximum" --dimensions "InstanceId=<your-instance-id>" --headers
```

Amazon CloudWatch returns the following (partial listing):

```
Time                  Maximum  Unit
        2011-01-09 23:18:00   0.38     Percent
        2011-01-09 23:24:00   0.38     Percent
        2011-01-09 23:30:00   0.38     Percent
        2011-01-09 23:36:00   0.38     Percent
        2011-01-09 23:42:00   0.38     Percent
        2011-01-09 23:48:00   0.39     Percent
        2011-01-09 23:54:00   0.38     Percent
        2011-01-10 00:00:00   0.38     Percent
        2011-01-10 00:06:00   0.38     Percent
        2011-01-10 00:12:00   0.38     Percent
```

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

Query API

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

- Call `GetMetricStatistics` with the following parameters:
  - *`MetricName`* = `CPUUtilization`
  - *`Period`* = `3600`
  - *`Statistics`* list includes `Maximum`
  - *`Dimensions`* (*`Name`*=`InstanceId`, *`Value`*="*`<your-instance-id>`*")
  - *`Namespace`* = `AWS/EC2`
  - *`StartTime`* = *`2011-01-09T23:18:00`*
  - *`EndTime`* = *`2011-01-12T23:18:00`*

Amazon CloudWatch Metrics, Namespaces, and Dimensions Reference

Topics

AWS Namespaces

All AWS services that provide Amazon CloudWatch data use a namespace string, beginning with "AWS/". The following services push metric datapoints to CloudWatch.

AWS Product	Namespace
AWS Billing	`AWS/Billing`
Amazon DynamoDB	`AWS/DynamoDB`
Amazon ElastiCache	`AWS/ElastiCache`
Amazon Elastic Block Store	`AWS/EBS`
Amazon Elastic Compute Cloud	`AWS/EC2`
Amazon Elastic MapReduce	`AWS/EMR`
Amazon Relational Database	`AWS/RDS`
Amazon Simple Notification Service	`AWS/SNS`
Amazon Simple Queue Service	`AWS/SQS`
Amazon Storage Gateway	`AWS/StorageGateway`
Auto Scaling	`AWS/AutoScaling`
Elastic Load Balancing	`AWS/ELB`

AWS Billing Dimensions and Metrics

Metric	Description
`EstimatedCharges`	The estimated charges for your AWS usage. This can either be estimated charges for one service or a roll-up of estimated charges for all services.

Dimensions for AWS Billing Metrics

AWS Billing sends the ServiceName and LinkedAccount dimensions to Amazon CloudWatch.

Dimension	Description
`ServiceName`	The name of the AWS service. This dimension is omitted for the total of estimated charges across all services.

Dimension	Description
`LinkedAccount`	The linked account number. This is used for consolidated billing only. This dimension is omitted for thte total of all accounts.

Amazon DynamoDB Dimensions and Metrics

Amazon DynamoDB Dimensions and Metrics

The following metrics are available from the Amazon DynamoDB Service. The service only sends metrics when they have a non-zero value. For example, if no requests generating a 400 status code occur in a time period, you would see no data for the UserErrors metric that reports requests generating a 400 status code.

Note

The Statistic values available through Amazon CloudWatch, such as *`Average`* or *`Sum`*, are not always applicable to every metric. However, they are all available through the console, API, and command line client for all services. For each metric, be aware of the list of Valid Statistics for the Amazon DynamoDB metrics to track useful information. For example, Amazon CloudWatch can monitor each time an Amazon DynamoDB request is refused (the `ThrottledRequests` metric). It marks that event as one occurrence. If the request is retried and also refused, Amazon CloudWatch marks the second event as one occurrence, too. The *`Sum`* statistic is now `2`. But, the *`Average`* statistic for the `ThrottledRequests` metric is simply `1`, if a request is throttled in the specified time period, once or repeatedly. For the `ThrottledRequests` metric, use the listed Valid Statistics (either Sum or SampleCount) to see the trend of `ThrottledRequests` over a specified time period.

Metric	Description
`SuccessfulRequestLatency`	The number of successful requests in the specified time period. By default, `SuccessfulRequestLatency` provides the elapsed time for successful calls. You can see statistics for the Minimum, Maximum, or Average, over time. **Note** Cloudwatch also provides a SampleCount statistic: the total number of successful calls for a sample time period. View (namespace): `AWS/DynamoDB, TableName, Operation` Units: Milliseconds (or a count for SampleCount) Valid Statistics: Minimum, Maximum, Average, SampleCount
`UserErrors`	The number of requests generating a 400 status code (likely indicating a client error) response in the specified time period. View (namespace): `All Metrics` Units: Count Valid Statistics: Sum, SampleCount

Metric	Description
`SystemErrors`	The number of requests generating a 500 status code (likely indicating a server error) response in the specified time period. View (namespace): `AWS/DynamoDB, TableName` Units: Count Valid Statistics: Sum, SampleCount
`ThrottledRequests`	The number of user requests that exceeded the preset provisioned throughput limits in the specified time period. View (namespace): `AWS/DynamoDB, TableName` Units: Count Valid Statistics: Sum, SampleCount
`ConsumedReadCapacityUnits`	The amount of read capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. For more information, see Provisioned Throughput in Amazon DynamoDB. View (namespace): `AWS/DynamoDB, TableName` **Note** Use the Sum value to calculate the provisioned throughput. For example, get the Sum value over a span of 5 minutes. Divide the Sum value by the number of seconds in 5 minutes (300) to get an average for the ConsumedReadCapacityUnits per second. You can compare the calculated value to the provisioned throughput value you provide Amazon DynamoDB. View (namespace): `AWS/DynamoDB, TableName` Units: Count Valid Statistics: Minimum, Maximum, Average, Sum

Metric	Description
`ConsumedWriteCapacityUnits`	The amount of write capacity units consumed over the specified time period, so you can track how much of your provisioned throughput is used. For more information, see Provisioned Throughput in Amazon DynamoDB. **Note** Use the Sum value to calculate the provisioned throughput. For example, get the Sum value over a span of 5 minutes. Divide the Sum value by the number of seconds in 5 minutes (300) to get an average for the ConsumedWriteCapacityUnits per second. You can compare the calculated value to the provisioned throughput value you provide Amazon DynamoDB. View (namespace): `AWS/DynamoDB, TableName` Units: Count Valid Statistics: Minimum, Maximum, Average, Sum
`ReturnedItemCount`	The number of items returned by a Scan or Query operation. View (namespace): `AWS/DynamoDB, TableName` Units: Count Valid Statistics: Minimum, Maximum, Average, SampleCount, Sum

Dimensions for Amazon DynamoDB Metrics

The metrics for Amazon DynamoDB are qualified by the values for the account, table name, or operation. Account level metrics display when you select **AWS/DynamoDB** as the viewing option. Otherwise, Amazon DynamoDB data can be retrieved along any of the following dimensions in the table below. Some metrics allow you to specify both a table name and operation, depending on the viewing option you specify.

Dimension	Description
`TableName`	This dimension limits the data you request to a specific table. This value can be any table name for the current account.

Dimension	Description
`Operation`	The operation corresponds to the Amazon DynamoDB service API, and can be one of the following: • `PutItem` • `DeleteItem` • `UpdateItem` • `GetItem` • `BatchGetItem` • `Scan` • `Query` For all of the operations in the current Amazon DynamoDB service API, see Operations in Amazon DynamoDB.

Amazon ElastiCache Dimensions and Metrics

Metric Dimensions

All Amazon ElastiCache metrics use the "AWS/ElastiCache" namespace and provide metrics for a single dimension, the *CacheNodeId*, which is the automatically-generated identifier for each Cache Node in the Cache Cluster. You can find out what these values are for your Cache Nodes using the `DescribeCacheClusters` API or **elasticache-describe-cache-clusters** command line utility.

Each metric is published under a single set of dimensions. When retrieving metrics, you must supply both the CacheClusterId and CacheNodeId dimensions.

Available Metrics

Amazon ElastiCache provides both host-level metrics (for example, CPU usage) and Memcached-specific metrics (i.e. number of gets). These metrics are measured and published for each Cache node in 60-second intervals.

The following table lists Memcached-specific metrics provided by Amazon ElastiCache at the Cache Node level.

Metric	Description	Unit
CPUUtilization	The percentage of CPU utilization.	Percent
SwapUsage	The amount of swap used on the host.	Bytes
FreeableMemory	The amount of free memory available on the host.	Bytes
NetworkBytesIn	The number of bytes the host has read from the network.	Bytes
NetworkBytesOut	The number of bytes the host has written to the network.	Bytes

The following table lists the cache node-level metrics provided by Amazon ElastiCache that are derived from the Memcached stats command.

Note

For complete documentation of the Memcached stats command, go to https://github.com/memcached/memcached/blob/master/doc/protocol.txt.

Metric	Description	Unit
BytesUsedForCacheItems	The number of bytes used to store cache items.	Bytes
BytesReadIntoMemcached	The number of bytes the cache has written to client connections.	Bytes
CasBadval	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
CasHits	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
CasMisses	The number of Cas requests the cache has received where the key requested was not found.	Count
CmdFlush	The number of flush commands the cache has received.	Count
CmdGet	The number of get commands the cache has received.	Count
CmdSet	The number of set commands the cache has received.	Count
CPUUtilization	The percentage of CPU utilization.	Percent
CurrConnections	A count of the number of connections connected to the cache at an instant in time. Note that due to the design of Memcached, this will always return a minimum count of 10.	Count
CurrItems	A count of the number of items currently stored in the cache.	Count
DecrHits	The number of decrement requests the cache has received where the requested key was found.	Count
DecrMisses	The number of decrement requests the cache has received where the requested key was not found.	Count
DeleteHits	The number of delete requests the cache has received where the requested key was found.	Count
DeleteMisses	The number of delete requests the cache has received where the requested key was not found.	Count
Evictions	The number of non-expired items the cache evicted to allow space for new writes.	Count

Metric	Description	Unit
GetHits	The number of get requests the cache has received where the key requested was found.	Count
GetMisses	The number of get requests the cache has received where the key requested was not found.	Count
IncrHits	The number of increment requests the cache has received where the key requested was found.	Count
IncrMisses	The number of increment requests the cache has received where the key requested was not found.	Count
Reclaimed	The number of expired items the cache evicted to allow space for new writes.	Count

The following table describes the available calculated cache level metrics.

Metric	Description	Unit
NewConnections	The number of new connections the cache has received. This is derived from the memcached total_connections statistic by recording the change in total_connections across a period of time. This will always be at least 1, due to a connection reserved for a Amazon ElastiCache.	Count
NewItems	The number of new items the cache has stored. This is derived from the memcached total_items statistic by recording the change in total_items across a period of time.	Count
UnusedMemory	The amount of unused memory the cache can use to store items. This is derived from the memcached statistics limit_maxbytes and bytes by subtracting bytes from limit_maxbytes.	Bytes

Amazon EBS Dimensions and Metrics

Amazon Elastic Block Store sends data points to Amazon CloudWatch for several metrics. All mounted Amazon EBS volumes automatically send five-minute metrics to Amazon CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon EBS volumes.

Amazon EBS Metrics

You can use the Amazon CloudWatch `GetMetricStatistics` API to get any of the Amazon EBS volume metrics listed in the following table. The period for all the metrics is 5 minutes, which means the system reports one data point every 5 minutes for each metric for each volume, and that data point covers the volume's previous 5 minutes of activity.

The following table groups metrics that are similar. The metrics in the first two rows are also available for the local stores on Amazon EC2 instances.

Metric	Description

`VolumeReadBytes`
 	
`VolumeWriteBytes`
	The total number of bytes transferred in the period.
	
	Units: Bytes

`VolumeReadOps`
 	
`VolumeWriteOps`
	The total number of operations in the period.
	
	Units: Count

`VolumeTotalReadTime`
 	
`VolumeTotalWriteTime`
	The total number of seconds spent by all operations that completed in the period. If multiple requests are submitted at the same time, this total could be greater than the length of the period. For example, say the period is 5 minutes (300 seconds); if 700 operations completed during that period, and each operation took 1 second, the value would be 700 seconds.
	
	Units: Seconds

`VolumeIdleTime`	The total number of seconds in the period when no read or write operations were submitted. Units: Seconds
`VolumeQueueLength`	The number of read and write operation requests waiting to be completed in the period. Units: Count

Dimensions for Amazon EBS Metrics

The only dimension that Amazon EBS sends to Amazon CloudWatch is the Volume ID. This means that all available statistics are filtered by Volume ID.

Amazon Elastic Compute Cloud Dimensions and Metrics

Topics

This section discusses the metrics and dimensions that Amazon Elastic Compute Cloud (Amazon EC2) sends to Amazon CloudWatch, and describes how to enable detailed (one-minute) monitoring for an EC2 instance. Amazon CloudWatch offers basic (five-minute) monitoring for Amazon EC2 by default. To access detailed monitoring of Amazon EC2 instances, you must enable it.

Amazon EC2 Metrics

The following metrics are available from each EC2 instance.

Metric	Description
`CPUUtilization`	The percentage of allocated EC2 compute units that are currently in use on the instance. This metric identifies the processing power required to run an application upon a selected instance. Units: *Percent*

Metric	Description
`DiskReadOps`	Completed read operations from all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 43).) This metric identifies the rate at which an application reads a disk. This can be used to determine the speed in which an application reads data from a hard disk. Units: *Count*
`DiskWriteOps`	Completed write operations to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 43).) This metric identifies the rate at which an application writes to a hard disk. This can be used to determine the speed in which an application saves data to a hard disk. Units: *Count*
`DiskReadBytes`	Bytes read from all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 43).) This metric is used to determine the volume of the data the application reads from the hard disk of the instance. This can be used to determine the speed of the application. Units: *Bytes*
`DiskWriteBytes`	Bytes written to all ephemeral disks available to the instance (if your instance uses Amazon EBS, see Amazon EBS Metrics (p. 43).) This metric is used to determine the volume of the data the application writes onto the hard disk of the instance. This can be used to determine the speed of the application. Units: *Bytes*
`NetworkIn`	The number of bytes received on all network interfaces by the instance. This metric identifies the volume of incoming network traffic to an application on a single instance. Units: *Bytes*
`NetworkOut`	The number of bytes sent out on all network interfaces by the instance. This metric identifies the volume of outgoing network traffic to an application on a single instance. Units: *Bytes*

Metric	Description
`StatusCheckFailed`	A combination of StatusCheckFailed_Instance and StatusCheckFailed_System that reports if either of the status checks has failed. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure. **Note** Status check metrics are available at 5 minute frequency and are not available in Detailed Monitoring. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state. Units: *Count*
`StatusCheckFailed_Instance`	Reports whether the instance has passed the EC2 instance status check in the last 5 minutes. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure. **Note** Status check metrics are available at 5 minute frequency and are not available in Detailed Monitoring. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state. Units: *Count*
`StatusCheckFailed_System`	Reports whether the instance has passed the EC2 system status check in the last 5 minutes. Values for this metric are either 0 (zero) or 1 (one.) A zero indicates that the status check passed. A one indicates a status check failure. **Note** Status check metrics are available at 5 minute frequency and are not available in Detailed Monitoring. For a newly launched instance, status check metric data will only be available after the instance has completed the initialization state. Status check metrics will become available within a few minutes of being in the running state. Units: *Count*

Amazon CloudWatch data for a new EC2 instance typically becomes available within one minute of the end of the first period of time requested (the *aggregation period*) in the query. You can set the period—the length of time over which statistics are aggregated—with the Period parameter. For more information on periods, see Periods (p. 8).

You can use the currently available dimensions for EC2 instances (for example, `ImageID` or `InstanceType`) to refine the metrics returned. For information about the dimensions you can use with EC2, see Dimensions for Amazon EC2 Metrics (p. 47).

Dimensions for Amazon EC2 Metrics

EC2 instance data can be filtered using any of the dimensions in the following table.

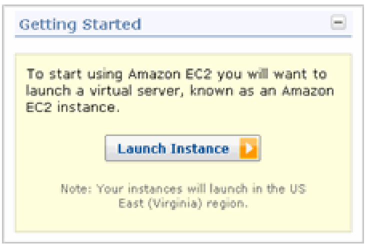
Dimension	Description
`AutoScalingGroupName`	This dimension filters the data you request for all instances in a specified capacity group. An AutoScalingGroup is a collection of instances you define if you're using the Auto Scaling service. This dimension is available only for EC2 metrics when the instances are in such an AutoScalingGroup. Available for instances with Detailed or Basic Monitoring enabled.
`ImageId`	This dimension filters the data you request for all instances running this EC2 Amazon Machine Image (AMI). Available for instances with Detailed Monitoring enabled.
`InstanceId`	This dimension filters the data you request for the identified instance only. This helps you pinpoint an exact instance from which to monitor data. Available for instances with Detailed Monitoring enabled.
`InstanceType`	This dimension filters the data you request for all instances running with this specified instance type. This helps you categorize your data by the type of instance running. For example, you might compare data from an m1.small instance and an m1.large instance to determine which has the better business value for your application. Available for instances with Detailed Monitoring enabled.

Activating Detailed Monitoring for Amazon EC2

The following procedure walks through the steps to enable detailed metric collection for an EC2 instance.

To activate detailed metrics through the console

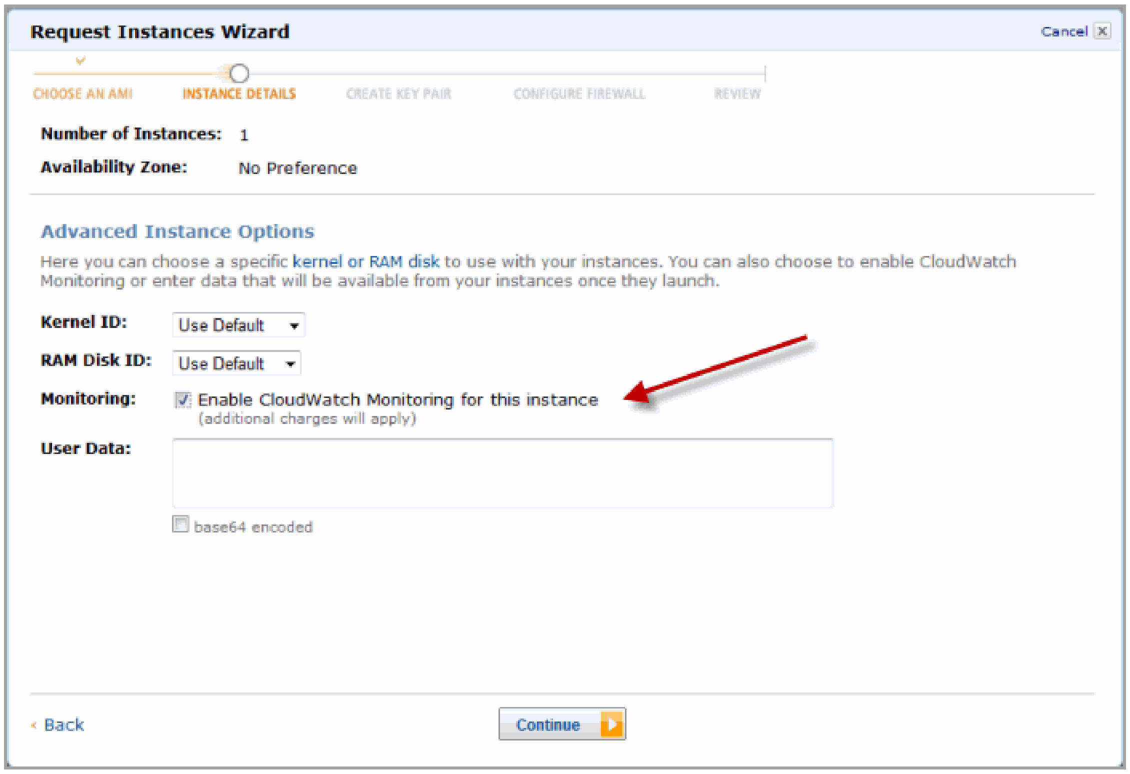
1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Click the **Launch Instance** button.


Getting Started

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (Virginia) region.

3. Select an AMI from the list to display the **Request Instances Wizard** dialog box and configure your EC2 instance.
4. On the next step of the **Request Instances Wizard**, click the **Enable CloudWatch monitoring for this instance** check box.



Number of Instances: 1
Availability Zone: No Preference

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use with your instances. You can also choose to enable CloudWatch Monitoring or enter data that will be available from your instances once they launch.

Kernel ID: Use Default
RAM Disk ID: Use Default
Monitoring: [✓] Enable CloudWatch Monitoring for this instance
(additional charges will apply)
User Data:
☐ base64 encoded

5. Continue through the remaining steps of the **Request Instances Wizard** and click the **Launch** button.

The instance you launched has detailed monitoring enabled.

Amazon Elastic MapReduce Dimensions and Metrics

This section discusses the metrics and dimensions that Amazon Elastic MapReduce (Amazon EMR) sends to Amazon CloudWatch. All Amazon EMR job flows automatically send metrics in five-minute intervals. Metrics are archived for two weeks; after that period, the data is discarded.

Amazon EMR Metrics

Amazon EMR sends the following metrics to Amazon CloudWatch.

Note

Amazon EMR pulls metrics from a job flow. If a job flow becomes unreachable, no metrics will be reported until the job flow becomes available again.

Metric	Description
`CoreNodesPending`	The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use Case: Monitor job flow health Units: *Count*

Metric	Description
`CoreNodesRunning`	The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use Case: Monitor job flow health Units: *Count*
`HBaseBackupFailed`	Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase job flows. Use Case: Monitor HBase backups Units: *Count*
`HBaseMostRecentBackupDuration`	The amount of time it took the previous backup to complete. This metric is set regardless of whether the last comppleted backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes since the backup started. This metric is only reported for HBase job flows. Use Case: Monitor HBase Backups Units: *Minutes*
`HBaseTimeSinceLastSuccessfulBackup`	The number of elapsed minutes since the last successful HBase backup started on your cluster. This metric is only reported for HBase job flows. Use Case: Monitor HBase backups Units: *Minutes*
`HDFSBytesRead`	The number of bytes read from HDFS. Use Case: Analyze job flow performance, Monitor job flow progress Units: *Count*
`HDFSBytesWritten`	The number of bytes written to HDFS. Use Case: Analyze job flow performance, Monitor job flow progress Units: *Count*
`HDFSUtilization`	The percentage of HDFS storage currently used. Use Case: Analyze job flow performance Units: *Percent*

Metric	Description
`IsIdle`	Indicates that a job flow is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the job flow was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer. Use Case: Monitor job flow performance Units: *Count*
`JobsFailed`	The number of jobs in the job flow that have failed. Use Case: Monitor job flow health Units: *Count*
`JobsRunning`	The number of jobs in the job flow that are currently running. Use Case: Monitor job flow health Units: *Count*
`LiveDataNodes`	The percentage of data nodes that are receiving work from Hadoop. Use Case: Monitor job flow health Units: *Percent*
`LiveTaskTrackers`	The percentage of task trackers that are functional. Use Case: Monitor job flow health Units: *Percent*
`MapSlotsOpen`	The unused map task capacity. This is calculated as the maximum number of map tasks for a given job flow, less the total number of map tasks currently running in that job flow. Use Case: Analyze job flow performance Units: *Count*
`MissingBlocks`	The number of blocks in which HDFS has no replicas. These might be corrupt blocks. Use Case: Monitor job flow health Units: *Count*

Metric	Description
`ReduceSlotsOpen`	Unused reduce task capacity. This is calculated as the maximum reduce task capacity for a given job flow, less the number of reduce tasks currently running in that job flow. Use Case: Analyze job flow performance Units: *Count*
`RemainingMapTasks`	The number of remaining map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. Use Case: Monitor job flow progress Units: *Count*
`RemainingMapTasksPerSlot`	The ratio of the total map tasks remaining to the total map slots available in the cluster. Use Case: Analyze job flow performance Units: *Ratio*
`RemainingReduceTasks`	The number of remaining reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs will be generated. Use Case: Monitor job flow progress Units: *Count*
`RunningMapTasks`	The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs will be generated. Use Case: Monitor job flow progress Units: *Count*
`RunningReduceTasks`	The number of running reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs will be generated. Use Case: Monitor job flow progress Units: *Count*
`S3BytesRead`	The number of bytes read from Amazon S3. Use Case: Analyze job flow performance, Monitor job flow progress Units: *Count*

Metric	Description
`S3BytesWritten`	The number of bytes written to Amazon S3. Use Case: Analyze job flow performance, Monitor job flow progress Units: *Count*
`TaskNodesPending`	The number of core nodes waiting to be assigned. All of the task nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use Case: Monitor job flow health Units: *Count*
`TaskNodesRunning`	The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use Case: Monitor job flow health Units: *Count*
`TotalLoad`	The total number of concurrent data transfers. Use Case: Monitor job flow health Units: *Count*

Amazon EMR Metrics

The following dimensions are available for Amazon EMR.

Dimension	Description
JobFlowId	The identifier for a job flow. You can find this value by clicking on the job flow in the Amazon EMR console. It takes the form `j-`XXXXXXXXXXXX.
JobId	The identifier of a job within a job flow. You can use this to filter the metrics returned from a job flow down to those that apply to a single job within the job flow. JobId takes the form job_XXXXXXXXXXXX_XXXX.

Amazon RDS Dimensions and Metrics

Topics

This section discusses the metrics and dimensions that Amazon Relational Database Service sends to Amazon CloudWatch. Amazon CloudWatch provides detailed monitoring of Amazon RDS by default. Unlike Amazon EC2 and Auto Scaling, you do not need to specifically enable detailed monitoring.

Amazon RDS Metrics

The following metrics are available from Amazon Relational Database Service.

Metric	Description
`BinLogDiskUsage`	The amount of disk space occupied by binary logs on the master. Units: Bytes
`CPUUtilization`	The percentage of CPU utilization. Units: Percent
`DatabaseConnections`	The number of database connections in use. Units: Count
`FreeableMemory`	The amount of available random access memory. Units: Bytes
`FreeStorageSpace`	The amount of available storage space. Units: Bytes
`ReplicaLag`	The amount of time a Read Replica DB Instance lags behind the source DB Instance. Units: Seconds
`SwapUsage`	The amount of swap space used on the DB Instance. Units: Bytes
`ReadIOPS`	The average number of disk I/O operations per second. Units: Count/Second
`WriteIOPS`	The average number of disk I/O operations per second. Units: Count/Second
`ReadLatency`	The average amount of time taken per disk I/O operation. Units: Seconds
`WriteLatency`	The average amount of time taken per disk I/O operation. Units: Seconds
`ReadThroughput`	The average number of bytes read from disk per second. Units: Bytes/Second
`WriteThroughput`	The average number of bytes written to disk per second. Units: Bytes/Second

Dimensions for RDS Metrics

Amazon RDS data can be filtered along any of the following dimensions in the table below.

Dimension	Description
`DBInstanceIdentifier`	This dimension filters the data you request for a specific database instance.
`DatabaseClass`	This dimension filters the data you request for all instances in a database class. For example, you can aggregate metrics for all instances that belong to the database class `db.m1.small`
`EngineName`	This dimension filters the data you request for the identified engine name only. For example, you can aggregate metrics for all instances that have the engine name `mysql`.

Amazon SNS Dimensions and Metrics

Amazon SNS sends data points to Amazon CloudWatch for several metrics. All active topics automatically send five-minute metrics to Amazon CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon SNS. A topic stays active for six hours from the last activity (i.e. any API call) on the topic.

Amazon SNS Metrics

This section discusses the metrics that Amazon Simple Notification Service (Amazon SNS) sends to Amazon CloudWatch.

Metric	Description
`NumberOfMessagesPublished`	The number of messages published to the topic. Units: *Count* Valid Statistics: Sum
`PublishSize`	The size of messages published to the topic. Units: *Bytes* Valid Statistics: Minimum, Maximum, Average and Count
`NumberOfNotificationsDelivered`	The number of messages successfully delivered to all subscriptions of the topic. Units: *Count* Valid Statistics: Sum
`NumberOfNotificationsFailed`	The number of all notification attempts to the topic that failed delivery. Units: *Count* Valid Statistics: Sum

Dimensions for Amazon SNS Metrics

The only dimension that Amazon SNS sends to Amazon CloudWatch is TopicName. This means that all available statistics are filtered by TopicName.

Amazon SQS Dimensions and Metrics

Amazon SQS sends data points to Amazon CloudWatch for several metrics. All active queues automatically send five-minute metrics to Amazon CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for Amazon SQS. A queue stays active for six hours from the last activity (i.e. any API call) on the queue.

Amazon SQS Metrics

This section discusses the metrics that Amazon Simple Queue Service (Amazon SQS) sends to Amazon CloudWatch.

Metric	Description
`NumberOfMessagesSent`	The number of messages added to a queue. Units: *Count* Valid Statistics: Sum
`SentMessageSize`	The size of messages added to a queue. Units: *Bytes* Valid Statistics: Minimum, Maximum, Average and Count
`NumberOfMessagesReceived`	The number of messages returned by calls to the `ReceiveMessage` API action. Units: *Count* Valid Statistics: Sum
`NumberOfEmptyReceives`	The number of `ReceiveMessage` API calls that did not return a message. Units: *Count* Valid Statistics: Sum
`NumberOfMessagesDeleted`	The number of messages deleted from the queue. Units: *Count* Valid Statistics: Sum

Metric	Description
`ApproximateNumberOfMessagesDelayed`	The number of messages in the queue that are delayed and not available for reading immediately. This can happen when the queue is configured as a delay queue or when a message has been sent with a delay parameter. Units: *Count* Valid Statistics: Average
`ApproximateNumberOfMessagesVisible`	The number of messages available for retrieval from the queue. Units: *Count* Valid Statistics: Average
`ApproximateNumberOfMessagesNotVisible`	The number of messages that are in flight. Messages are considered in flight if they have been sent to a client but have not yet been deleted or have not yet reached the end of their visibility window. Units: *Count* Valid Statistics: Average

Dimensions for Amazon SQS Metrics

The only dimension that Amazon SQS sends to Amazon CloudWatch is QueueName. This means that all available statistics are filtered by QueueName.

AWS Storage Gateway Dimensions and Metrics

AWS Storage Gateway sends data points to Amazon CloudWatch for several metrics. All active queues automatically send five-minute metrics to Amazon CloudWatch. Detailed monitoring, or one-minute metrics, is currently unavailable for AWS Storage Gateway.

AWS Storage Gateway Metrics

The following metrics are available from the AWS Storage Gateway Service.

The following table describes the AWS Storage Gateway metrics that you can use to get information about your gateways. Specify the `GatewayId` or `GatewayName` dimension for each metric to view the data for a gateway.

Metric	Description
`ReadBytes`	The total number of bytes read from your on-premises applications in the reporting period for all volumes in the gateway. Use this metric to measure throughput by selecting the `Sum` statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the `Samples` statistic and dividing each data point by the Period. Units: Bytes
`WriteBytes`	The total number of bytes written to your on-premises applications in the reporting period for all volumes in the gateway. Use this metric to measure throughput by selecting the `Sum` statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the `Samples` statistic and dividing each data point by the Period. Units: Bytes
`ReadTime`	The total number of milliseconds spent to do reads from your on-premises applications in the reporting period for all volumes in the gateway. Use this metric with the `Average` statistic to measure average latency. Units: Milliseconds
`WriteTime`	The total number of milliseconds spent to do writes from your on-premises applications in the reporting period for all volumes in the gateway. Use this metric with the `Average` statistic to measure average latency. Units: Milliseconds
`QueuedWrites`	The number of bytes waiting to be written to AWS, sampled at the end of the reporting period for all volumes in the gateway. These bytes are kept in your gateway's working storage. Units: Bytes
`CloudBytesDownloaded`	The total number of bytes that the gateway downloaded from AWS during the reporting period. Use this metric to measure throughput by selecting the `Sum` statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the `Samples` statistic and dividing each data point by the Period. Units: Bytes
`CloudBytesUploaded`	The total number of bytes that the gateway uploaded to AWS during the reporting period. Use this metric to measure throughput by selecting the `Sum` statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the `Samples` statistic and dividing each data point by the Period. Units: Bytes

Metric	Description
`CloudDownloadLatency`	The total number of milliseconds spent reading data from AWS during the reporting period. Use this metric with the `Average` statistic to measure average latency. Units: Milliseconds
`WorkingStoragePercentUsed`	Percent utilization of the gateway's working storage. The sample is taken at the end of the reporting period. Units: Percent
`WorkingStorageUsed`	The total number of bytes being used in the gateway's working storage. The sample is taken at the end of the reporting period. Units: Bytes
`WorkingStorageFree`	The total amount of unused space in the gateway's working storage. The sample is taken at the end of the reporting period. Units: Bytes

The following table describes the AWS Storage Gateway metrics that you can use to get information about your storage volumes. Specify the `VolumeId` dimension for each metric to view the data for a storage volume.

Metric	Description
`ReadBytes`	The total number of bytes read from your on-premises applications in the reporting period. Use this metric to measure throughput by selecting the `Sum` statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the `Samples` statistic and dividing each data point by the Period. Units: Bytes
`WriteBytes`	The total number of bytes written to your on-premises applications in the reporting period. Use this metric to measure throughput by selecting the `Sum` statistic and dividing by the Period. Use this metric to measure operations rate (IOPS) by selecting the `Samples` statistic and dividing each data point by the Period. Units: Bytes
`ReadTime`	The total number of milliseconds spent to do reads from your on-premises applications in the reporting period. Use this metric with the `Average` statistic to measure average latency. Units: Milliseconds

Metric	Description
`WriteTime`	The total number of milliseconds spent to do writes from your on-premises applications in the reporting period. Use this metric with the `Average` statistic to measure average latency. Units: Milliseconds
`QueuedWrites`	The number of bytes waiting to be written to AWS, sampled at the end of the reporting period. Units: Bytes

Dimensions for AWS Storage Gateway Metrics

The Amazon CloudWatch namespace for the service is `AWS/StorageGateway`. Data is available automatically in 5-minute periods at no charge.

Dimension	Description
`GatewayId`, `GatewayName`	These dimensions filter the data you request to gateway-specific metrics. You can identify a gateway to work by its `GatewayId` or its `GatewayName`. However, note that if the name of your gateway was changed for the time range that you are interested in viewing metrics, then you should use the `GatewayId`. Throughput and latency data of a gateway is based on all the volumes for the gateway. For information about working with gateway metrics, see Measuring Performance Between Your Gateway and AWS.
`VolumeId`	This dimension filters the data you request to volume-specific metrics. Identify a storage volume to work with by its `VolumeId`. For information about working with volume metrics, see Measuring Performance Between Your Application and Gateway.

Auto Scaling Dimensions and Metrics

Topics

This section discusses the metrics that Auto Scaling instances and groups send to Amazon CloudWatch and describes how to enable detailed (one-minute) monitoring and basic (five-minute) monitoring.

Auto Scaling Instance Support

This section discusses the metrics that Auto Scaling instances send to Amazon CloudWatch. Instance metrics are the metrics that an individual Amazon EC2 instance sends to Amazon CloudWatch. Instance metrics are the same metrics available for any Amazon EC2 instance, whether or not it is in an Auto Scaling group.

Amazon CloudWatch offers basic or detailed monitoring. Basic monitoring sends aggregated data about each instance to Amazon CloudWatch every five minutes. Detailed monitoring offers more frequent aggregated data by sending data from each instance every minute.

Note

Selecting detailed monitoring is a prerequisite for the collection of Auto Scaling group metrics. For more information, see Auto Scaling Group Support (p. 61).

The following sections describe how to enable either detailed monitoring or basic monitoring.

Activating Detailed Instance Monitoring for Auto Scaling

To enable detailed instance monitoring for a new Auto Scaling group, you don't need to take any extra steps. One of your first steps when creating an Auto Scaling group is to create a launch configuration. Each launch configuration contains a flag named `InstanceMonitoring.Enabled`. The default value of this flag is `true`, so you don't need to set this flag if you want detailed monitoring.

If you have an Auto Scaling group for which you have explicitly selected basic monitoring, the switch to detailed monitoring involves several steps, especially if you have Amazon CloudWatch alarms configured to scale the group automatically.

To switch to detailed instance monitoring for an existing Auto Scaling group

1. Create a launch configuration that has the `InstanceMonitoring.Enabled` flag enabled. If you are using the command line tools, create a launch configuration with the `--monitoring-enabled` option.
2. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will enable detailed monitoring for new instances that it creates.
3. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

To...	Do This...
Preserve existing instances	Call `MonitorInstances` from the Amazon EC2 API for each existing instance to enable detailed monitoring.
Terminate existing instances	Call `TerminateInstanceInAutoScalingGroup` from the Auto Scaling API for each existing instance. Auto Scaling will use the updated launch configuration to create replacement instances with detailed monitoring enabled.

4. If you have Amazon CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the Amazon CloudWatch API to update each alarm so that the alarm's period value is set to 60 seconds.

Activating Basic Instance Monitoring for Auto Scaling

To create a new Auto Scaling group with basic monitoring instead of detailed monitoring, associate your new Auto Scaling group with a launch configuration that has the `InstanceMonitoring.Enabled` flag set to `false`. If you are using the command line tools, create a launch configuration with the `--monitoring-disabled` option.

To switch to basic instance monitoring for an existing Auto Scaling group

1. Create a launch configuration that has the `InstanceMonitoring.Enabled` flag disabled. If you are using the command line tools, create a launch configuration with the `--monitoring-disabled` option.

2. If you previously enabled group metrics with a call to `EnableMetricsCollection`, call `DisableMetricsCollection` on your Auto Scaling group to disable collection of all group metrics. For more information, see Auto Scaling Group Support (p. 61).
3. Call `UpdateAutoScalingGroup` to update your Auto Scaling group with the launch configuration you created in the previous step. Auto Scaling will disable detailed monitoring for new instances that it creates.
4. Choose one of the following actions to deal with all existing Amazon EC2 instances in the Auto Scaling group:

To...	Do This...
Preserve existing instances	Call `UnmonitorInstances` from the Amazon EC2 API for each existing instance to disable detailed monitoring.
Terminate existing instances	Call `TerminateInstanceInAutoScalingGroup` from the Auto Scaling API for each existing instance. Auto Scaling will use the updated launch configuration to create replacement instances with detailed monitoring disabled.

5. If you have Amazon CloudWatch alarms associated with your Auto Scaling group, call `PutMetricAlarm` from the Amazon CloudWatch API to update each alarm so that the alarm's period value is set to 300 seconds.

   Important

   If you do not update your alarms to match the five-minute data aggregations, your alarms will continue to check for statistics every minute and might find no data available for as many as four out of every five periods.

For more information on instance metrics for Amazon EC2 instances, see Amazon Elastic Compute Cloud Dimensions and Metrics (p. 44).

Auto Scaling Group Support

Group metrics are metrics that an Auto Scaling group sends to Amazon CloudWatch to describe the group rather than any of its instances. If you enable group metrics, Auto Scaling sends aggregated data to Amazon CloudWatch every minute. If you disable group metrics, Auto Scaling does not send any group metrics data to Amazon CloudWatch.

To enable group metrics

1. Enable detailed instance monitoring for the Auto Scaling group by setting the *`InstanceMonitoring.Enabled`* flag in the Auto Scaling group's launch configuration. For more information, see Auto Scaling Instance Support (p. 59).
2. Call `EnableMetricsCollection`, which is part of the Auto Scaling Query API. Alternatively, you can use the equivalent `as-enable-metrics-collection` command that is part of the Auto Scaling command line tools.

Auto Scaling group metrics table

You may enable or disable each of the following metrics, separately.

Metric	Description
`GroupMinSize`	The minimum size of the Auto Scaling group. Units: *Count*
`GroupMaxSize`	The maximum size of the Auto Scaling group. Units: *Count*
`GroupDesiredCapacity`	The number of instances that the Auto Scaling group attempts to maintain. Units: *Count*
`GroupInServiceInstances`	The number of instances that are running as part of the Auto Scaling group. This metric does not include instances that are pending or terminating. Units: *Count*
`GroupPendingInstances`	The number of instances that are pending. A pending instance is not yet in service. This metric does not include instances that are in service or terminating. Units: *Count*
`GroupTerminatingInstances`	The number of instances that are in the process of terminating. This metric does not include instances that are in service or pending. Units: *Count*
`GroupTotalInstances`	The total number of instances in the Auto Scaling group. This metric identifies the number of instances that are in service, pending, and terminating. Units: *Count*

Dimensions for Auto Scaling Group Metrics

The only dimension that Auto Scaling sends to Amazon CloudWatch is the name of the Auto Scaling group. This means that all available statistics are filtered by Auto Scaling group name.

Elastic Load Balancing Dimensions and Metrics

Topics

This section discusses the metrics and dimensions that Elastic Load Balancing sends to Amazon CloudWatch. Amazon CloudWatch provides detailed monitoring of Elastic Load Balancing by default. Unlike Amazon EC2, you do not need to specifically enable detailed monitoring.

Note

Elastic Load Balancing only emits Amazon CloudWatch metrics when requests are flowing through the load balancer.

Elastic Load Balancing Metrics

The following Elastic Load Balancing metrics are available from Amazon CloudWatch.

The HTTP response code metrics reflect the count of Elastic Load Balancing response codes that are sent to clients within a given time period. If no response codes in the category 2XX-5XX range are sent to clients within the given time period, values for these metrics will not be recorded in CloudWatch.

Metric	Description
`Latency`	Time elapsed after the load balancer receives a request until it receives the corresponding response. Units: Seconds Valid Statistics: Minimum, Maximum, Average, and Count
`RequestCount`	The number of requests handled by the load balancer. Units: Count Valid Statistics: Sum
`HealthyHostCount`	The number of healthy Amazon EC2 instances registered with the load balancer in a specified Availability Zone. Hosts that have not failed more health checks than the value of the unhealthy threshold are considered healthy. When evaluating this metric, the dimensions must be provided for *`LoadBalancerName`* and *`AvailabilityZone`*. The metric represents the count of healthy instances in the specified Availability Zone. Instances may become unhealthy due to connectivity issues, health checks returning non-200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check. To get the total count of all healthy hosts, this metric must be retrieved for each registered Availability Zone and then all the metrics need to be added together. Units: Count Valid Statistics: Minimum, Maximum, and Average
`UnHealthyHostCount`	The number of unhealthy Amazon EC2 instances registered with the load balancer in a specified Availability Zone. Hosts that have failed more health checks than the value of the unhealthy threshold are considered unhealthy. When evaluating this metric, the dimensions must be provided for *`LoadBalancerName`* and *`AvailabilityZone`*. The metric represents the count of unhealthy instances in the specified Availability Zone. Instances may become unhealthy due to connectivity issues, health checks returning non-200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check. To get the total count of all unhealthy hosts, this metric must be retrieved for each registered Availability Zone and then all the metrics need to be added together. Units: Count Valid Statistics: Minimum, Maximum, and Average

Metric	Description
`HTTPCode_ELB_4XX`	Count of HTTP response codes generated by Elastic Load Balancing that are in the 4xx (client error) series. Units: Count Valid Statistics: Sum
`HTTPCode_ELB_5XX`	Count of HTTP response codes generated by Elastic Load Balancing that are in the 5xx (server error) series. Elastic Load Balancing may generate 5xx errors if no back-end instances are registered, no healthy back-end instances, or the request rate exceeds Elastic Load Balancing's current available capacity. Units: Count Valid Statistics: Sum
HTTPCode_Backend_2XX	Count of HTTP response codes generated by back-end instances that are in the 2xx (success) series. Units: Count Valid Statistics: Sum
HTTPCode_Backend_3XX	Count of HTTP response codes generated by back-end instances that are in the 3xx (user action required) series. Units: Count Valid Statistics: Sum
HTTPCode_Backend_4XX	Count of HTTP response codes generated by back-end instances that are in the 4xx (client error) series. This response count does not include any responses that were generated by Elastic Load Balancing. Units: Count Valid Statistics: Sum
HTTPCode_Backend_5XX	Count of HTTP response codes generated by back-end instances that are in the 5xx (server error) series. This response count does not include any responses that were generated by Elastic Load Balancing. Units: Count Valid Statistics: Sum

Dimensions for Elastic Load Balancing Metrics

You can use the currently available dimensions for Elastic Load Balancing to refine the metrics returned by a query. For example, you could use *HealthyHostCount* and dimensions *LoadBalancerName* and *AvailabilityZone* to get the Average number of healthy Instances behind the specified LoadBalancer within the specified Availability Zone for a given period of time.

Elastic Load Balancing data can be aggregated along any of the following dimensions shown in the table below.

Dimension	Description
`LoadBalancerName`	Limits the metric data to Amazon EC2 instances that are connected to the specified load balancer.
`AvailabilityZone`	Limits the metric data to load balancers in the specified Availability Zone.

Creating Amazon CloudWatch Alarms

Topics

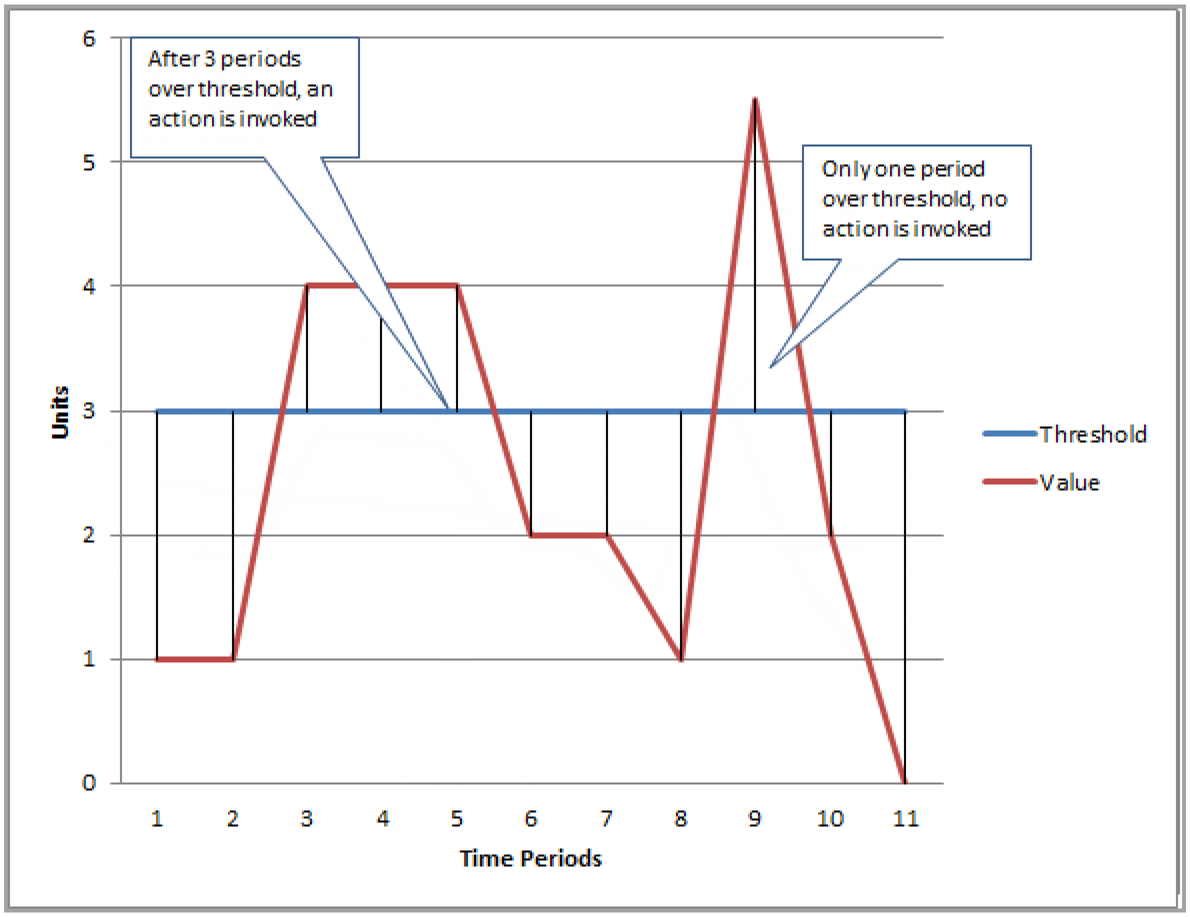
You can create an Amazon CloudWatch alarm that sends an Amazon Simple Notification Service message when the alarm changes state. An alarm watches a single metric over a time period you specify, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or Auto Scaling policy. Alarms invoke actions for sustained state changes only. Amazon CloudWatch alarms will not invoke actions simply because they are in a particular state, the state must have changed and been maintained for a specified number of periods.

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

An alarm has three possible states:

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

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



Note

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

Common Features of Alarms

- You can create up to 400 alarms per AWS account. To create or update an alarm, you use the `PutMetricAlarm` API function (`mon-put-metric-alarm` command).
- You can list any or all of the currently configured alarms, and list any alarms in a particular state using the `DescribeAlarms` API (`mon-describe-alarms` command). You can further filter the list by time range.
- You can disable and enable alarms by using the `DisableAlarmActions` and `EnableAlarmActions` APIs (`mon-disable-alarm-actions` and `mon-enable-alarm-actions` commands).
- You can test an alarm by setting it to any state using the `SetAlarmState` API (`mon-set-alarm-state` command). This temporary state change lasts only until the next alarm comparison occurs.
- Finally, you can view an alarm's history using the `DescribeAlarmHistory` API (`mon-describe-alarm-history` command). Amazon CloudWatch preserves alarm history for two weeks. Each state transition is marked with a unique time stamp. In rare cases, your history might show more than one notification for a state change. The time stamp enables you to confirm unique state changes.

Set Up Amazon SNS

Topics

This section shows you how to set up an Amazon SNS topic to send an email notification when an alarm changes state. For information about Amazon SNS and instructions on how to create an Amazon SNS email notification, go to the Amazon Simple Notification Service Getting Started Guide.

Note

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

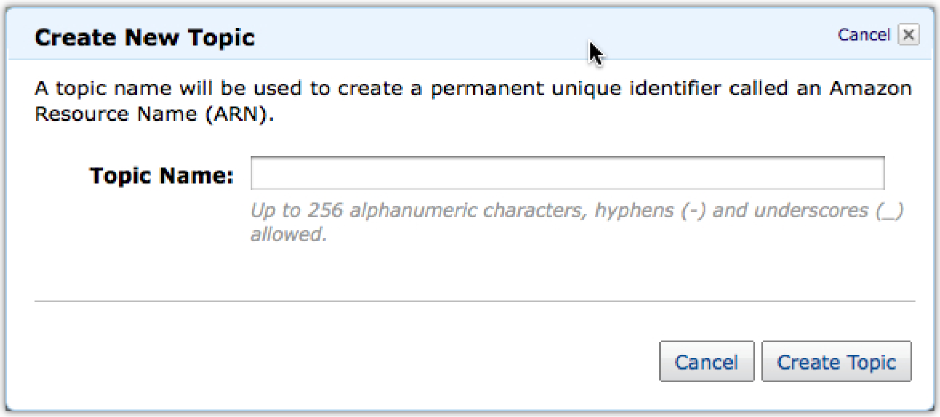
AWS Management Console

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

To create an Amazon SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. In **Getting Started**, click **Create New Topic**.

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



Create New Topic

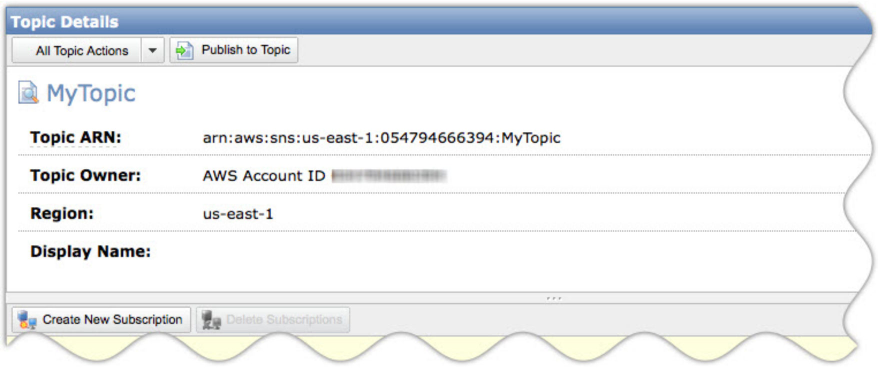
A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN).

Topic Name:

Up to 256 alphanumeric characters, hyphens (-) and underscores (_) allowed.

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

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



MyTopic

Topic ARN: arn:aws:sns:us-east-1:054794666394:MyTopic

Topic Owner: AWS Account ID [redacted]

Region: us-east-1

Display Name:

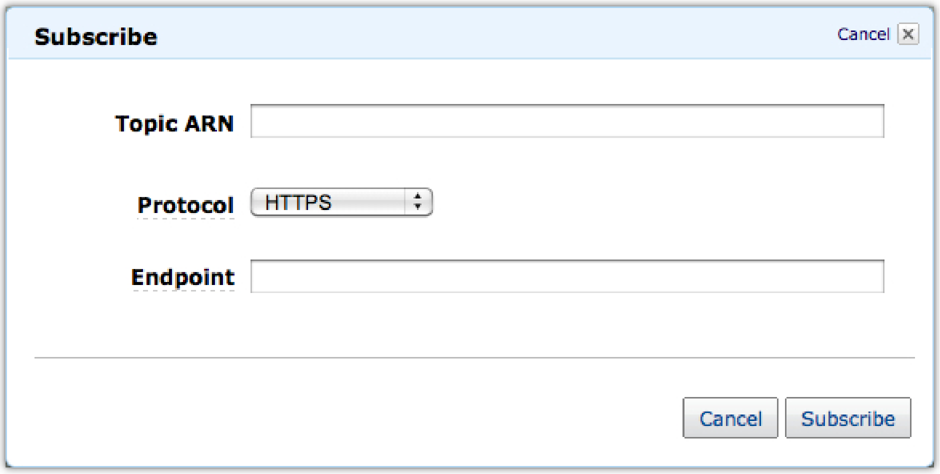
5. Copy the **Topic ARN** for the next task.

To subscribe to an Amazon SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. In the **Navigation** pane, click **My Subscriptions** in the **Navigation** pane.

   The **My Subscriptions** page opens.
3. Click **Create New Subscription**.

   The **Subscribe** dialog box opens.



Topic ARN

Protocol HTTPS

Endpoint

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

   Important

   Entourage Users: Entourage strips out the confirmation URL. Please enter an email address you can access in a different email application.
7. Go to your email application and open the message from AWS Notifications, and then click the link to confirm your subscription.

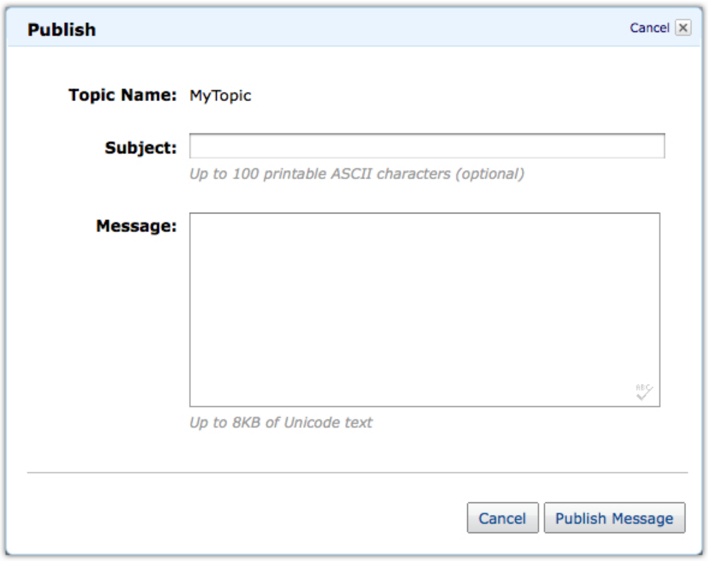
   Your web browser displays a confirmation response from Amazon SNS.

To publish to an Amazon SNS topic

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
2. In the **Navigation** pane, under **My Topics**, click the topic you want to publish.

   The **Topic Details** page opens.
3. Click **Publish to Topic**.

   The **Publish** dialog box opens.



Publish

Topic Name: MyTopic

Subject:

Up to 100 printable ASCII characters (optional)

Message:

Up to 8KB of Unicode text

4. Enter a subject line for your message in the **Subject** field, and a brief message in the **Message** field.
5. Click **Publish Message**.

   A confirmation dialog box opens.
6. Click **Close**.
7. Check your email to confirm that you received the message from the topic.

Command Line Tools

This scenario walks you through how to use the command line tools to create an Amazon SNS topic, and then publish a message directly to the topic to ensure that you have properly configured it.

Note

This scenario uses the Amazon SNS command line tools. To download the Amazon SNS command line tools, go to http://aws.amazon.com/developertools/3688.

To set up an Amazon SNS topic

1. Create the topic with the Amazon SNS CLI command `sns-create-topic`. You receive a topic resource name as a return value:

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

   Amazon SNS returns the following Topic ARN:

```
arn:aws:sns:us-east-1:887848922426:MyTopic
```

2. Subscribe your email address to the topic with the Amazon SNS CLI command `sns-subscribe`. You will receive a confirmation email message if the subscription request succeeds.

   ```
   Prompt>sns-subscribe  --topic arn:aws:sns:us-east-1:887848922426:MyTopic
   --protocol SMTP  --endpoint <your-email-address>
   ```

   Amazon SNS returns the following:

   ```
   Subscription request received
   ```

3. Confirm that you intend to receive email from Amazon SNS by clicking the confirmation link in the body of the message to complete the subscription process.
4. Check the subscription with the Amazon SNS CLI command `sns-list-subscription-by-topic`.

   ```
   Prompt>sns-list-subscription-by-topic  --topic arn:aws:sns:us-east-
   1:887848922426:MyTopic
   ```

   Amazon SNS returns the following:

   ```
   arn:aws:sns:us-east-1:887848922426:MyTopic:97fbacb0-7706-4ee9-9f8c-
   d56074b25278
     SMTP me@my_company.com
   ```

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

   ```
   Prompt>sns-publish  --message  "Verification"  --topic arn:aws:sns:us-east-
   1:887848922426:MyTopic
   ```

   Amazon SNS returns the following:

   ```
   b94635b4-9cc6-4895-b961-87e9a1ff8ff8
   ```

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

Send Email Based on CPU Usage Alarm

Topics

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

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

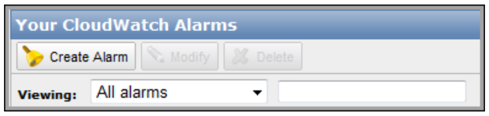
AWS Management Console

The **Create Alarm Wizard** steps you through the process of creating an alarm.

To open the Create Alarm Wizard

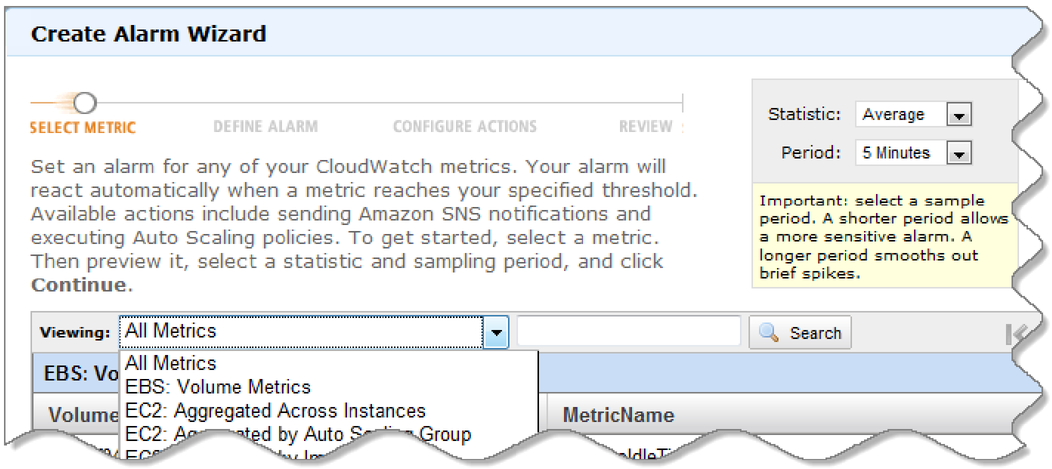
1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Alarms**.

   The **Your CloudWatch Alarms** page opens.

3. Click **Create Alarm**.

   The **SELECT METRIC** page of the **Create Alarm Wizard** opens.

   Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

   Statistic: Average
   Period: 5 Minutes

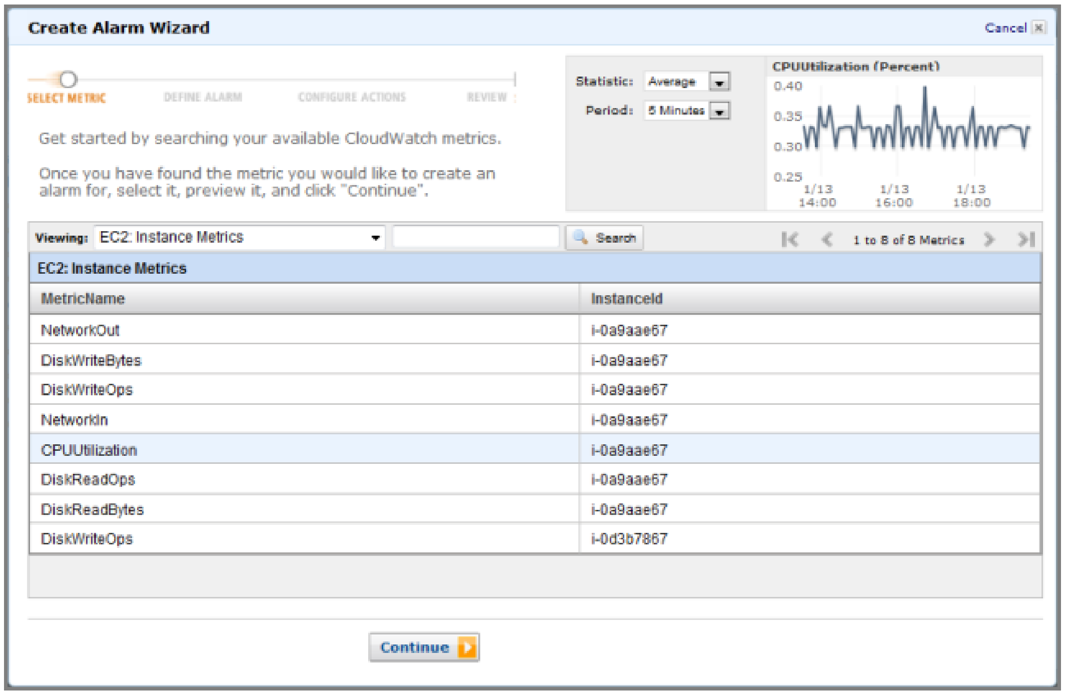
   Important: select a sample period. A shorter period allows a more sensitive alarm. A longer period smooths out brief spikes.

To select a metric for your alarm

1. In the **SELECT METRIC** page of the **Create Alarm Wizard**, select **EC2: Instance Metrics** from the **Viewing** drop-down list.

   The metrics available for individual instances appear in the **Metrics** pane.
2. Select a row that contains **CPUUtilization** for a specific instance ID.

   A graph showing average `CPUUtilization` for a single instance appears in the **Metrics** pane.



Get started by searching your available CloudWatch metrics.

Once you have found the metric you would like to create an alarm for, select it, preview it, and click "Continue".

Statistic: Average
Period: 5 Minutes

Viewing: EC2: Instance Metrics

1 to 8 of 8 Metrics

EC2: Instance Metrics

MetricName	InstanceId
NetworkOut	i-0a9aae67
DiskWriteBytes	i-0a9aae67
DiskWriteOps	i-0a9aae67
NetworkIn	i-0a9aae67
CPUUtilization	i-0a9aae67
DiskReadOps	i-0a9aae67
DiskReadBytes	i-0a9aae67
DiskWriteOps	i-0d3b7867

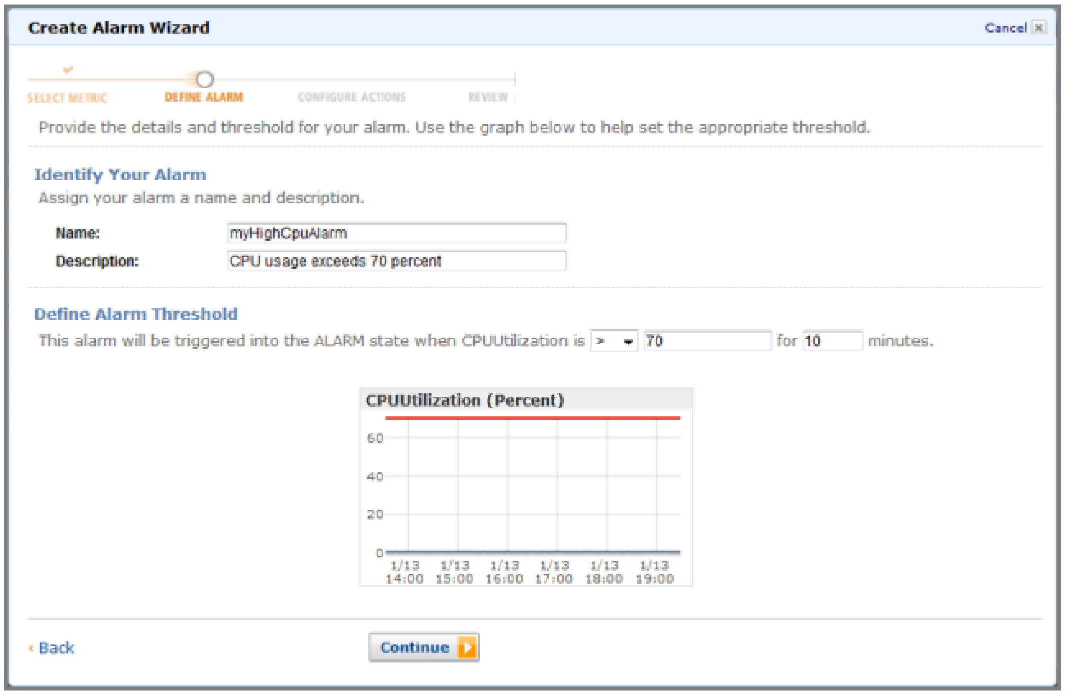
3. Select **Average** from the **Statistic** drop-down list.
4. Select a period from the **Period** drop-down list, for example: **`5 minutes`**.
5. Click **Continue**.

   The **DEFINE ALARM** page of the **Create Alarm Wizard** opens.

To define the alarm name, description, and threshold

1. In the **Name** field, enter the name of the alarm, for example: **`myHighCpuAlarm`**.
2. In the **Description** field, enter a description of the alarm, for example: **`CPU usage exceeds 70 percent.`**
3. Select **>** in the **Define Alarm Threshold** drop-down list.
4. Enter **`70`** in the first **Define Alarm Threshold** field and **`10`** in the second field.

   A graphical representation of the threshold appears on the page.



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

Identify Your Alarm

Assign your alarm a name and description.

Name: myHighCpuAlarm

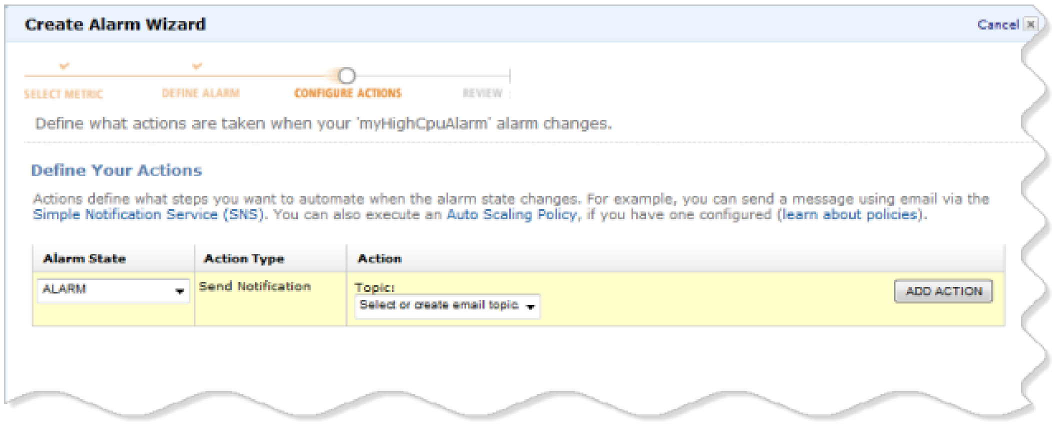
Description: CPU usage exceeds 70 percent

Define Alarm Threshold

This alarm will be triggered into the ALARM state when CPUUtilization is > 70 for 10 minutes.

5. Click **Continue**.

   The **CONFIGURE ACTIONS** page of the **Create Alarm Wizard** opens.



Define what actions are taken when your 'myHighCpuAlarm' alarm changes.

Define Your Actions

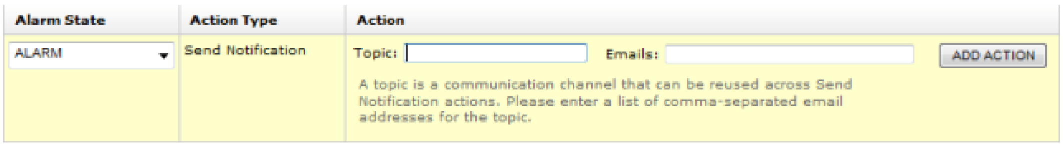
Actions define what steps you want to automate when the alarm state changes. For example, you can send a message using email via the Simple Notification Service (SNS). You can also execute an Auto Scaling Policy, if you have one configured (learn about policies).

Alarm State	Action Type	Action
ALARM	Send Notification	Topic: Select or create email topic

To configure an email action for an alarm

1. Select **ALARM** from the **Alarm State** drop-down list.
2. Select **Create Email Topic...** from the **Topic** drop-down list.

   Two new fields named **Topic** and **Emails** replace the **Topic** drop-down list.

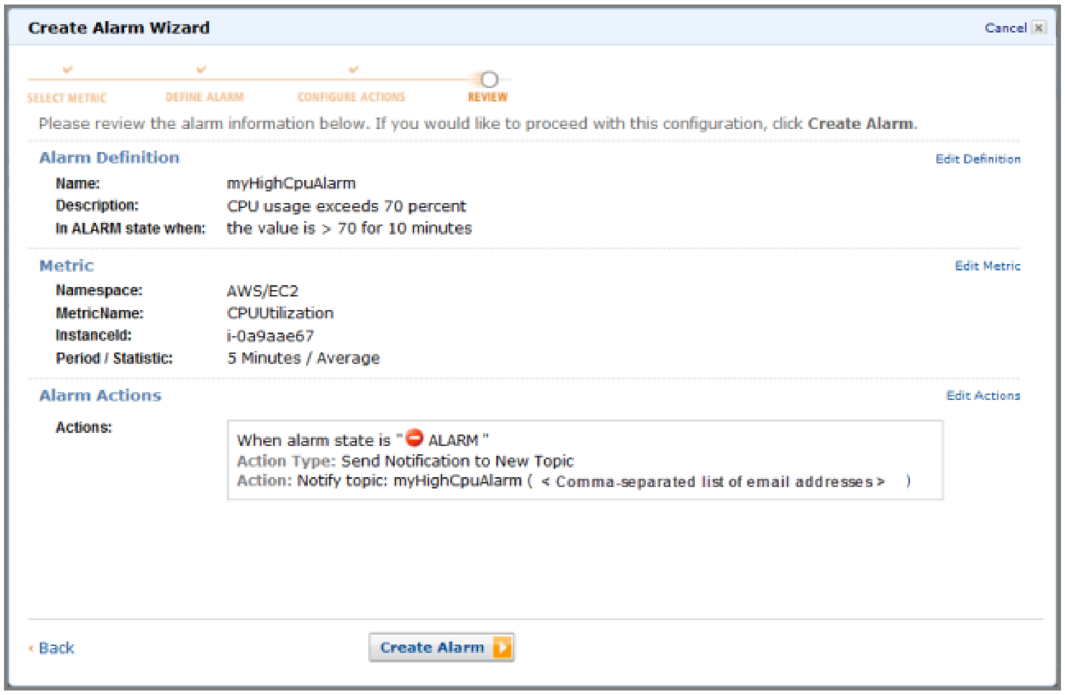


Alarm State	Action Type	Action
ALARM	Send Notification	Topic: Emails: A topic is a communication channel that can be reused across Send Notification actions. Please enter a list of comma-separated email addresses for the topic.

3. In the **Topic** field, enter a descriptive name for the Amazon SNS topic, for example: **`myHighCpuAlarm`**.
4. In the **Emails** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state.
5. Click **ADD ACTION**.

   The action is saved and the **ADD ACTION** button becomes a **REMOVE** button.
6. Click **Continue**.

   The **REVIEW** page of the **Create Alarm Wizard** opens.



Please review the alarm information below. If you would like to proceed with this configuration, click **Create Alarm**.

Alarm Definition — Edit Definition

Name:	myHighCpuAlarm
Description:	CPU usage exceeds 70 percent
In ALARM state when:	the value is > 70 for 10 minutes

Metric — Edit Metric

Namespace:	AWS/EC2
MetricName:	CPUUtilization
InstanceId:	i-0a9aae67
Period / Statistic:	5 Minutes / Average

Alarm Actions — Edit Actions

Actions:
When alarm state is " ALARM "
Action Type: Send Notification to New Topic
Action: Notify topic: myHighCpuAlarm (< Comma-separated list of email addresses >)

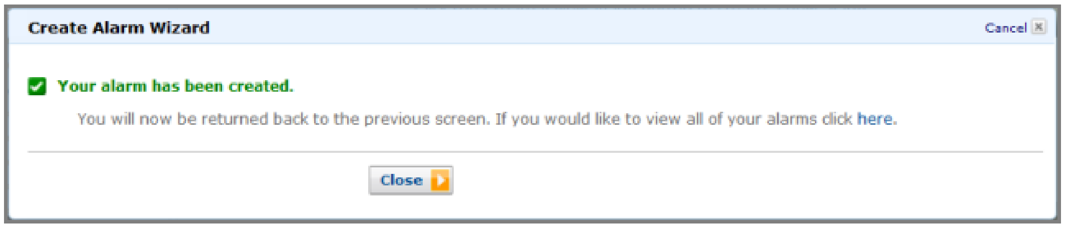
Now that you have defined the alarm and configured the alarm's actions, you are ready to review the settings and create the alarm.

To review the alarm settings and create the alarm

1. Review the alarm settings presented in the **REVIEW** page of the **Create Alarm Wizard**.

   You can make changes to the settings with the **Edit Definition**, **Edit Metric**, or **Edit Actions** links.
2. Click **Create Alarm** to complete the alarm creation process.

   A confirmation window opens.



Your alarm has been created.

You will now be returned back to the previous screen. If you would like to view all of your alarms click here.

3. Click **Close**.

Your alarm is created.

Command Line Tools

To send an Amazon SNS email message when CPU utilization exceeds 70 percent

1. Set up an Amazon SNS topic or retrieve the Topic Resource Name of the topic you intend to use. For help on setting up an Amazon SNS topic, see Set Up Amazon SNS (p. 67).
2. Create an alarm with the Amazon CloudWatch CLI command `mon-put-metric-alarm`. Use the values from the following example, but replace the values for `InstanceID` and `alarm-actions` with your own values.

   ```
   Prompt>mon-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 Great
   erThanThreshold  --dimensions  "InstanceId=i-12345678"  --evaluation-periods
    2 --alarm-actions arn:aws:sns:us-east-1:887848922426:MyTopic --unit Percent
   ```

   Amazon CloudWatch returns the following:

   ```
   OK-Created Alarm
   ```

3. Test the alarm by forcing an alarm state change with the Amazon CloudWatch CLI command `mon-set-alarm-state`.

   a. Change the alarm state from `INSUFFICIENT_DATA` to `OK`:

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

      Amazon CloudWatch returns the following:

      ```
      OK-Set alarm state value
      ```

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

      ```
      Prompt>mon-set-alarm-state cpu-mon --state-reason "testing" --state-value
       ALARM
      ```

      Amazon CloudWatch returns the following:

      ```
      OK-Set alarm state value
      ```

   c. Check that an email has been received.

Send Email Based on Load Balancer Alarm

Topics

- AWS Management Console (p. 76)
- Command Line Tools (p. 80)

This scenario walks you through how to use the AWS Management Console or the command line tools to set up an Amazon SNS notification and configure an alarm that monitors load balancer latency exceeding 100 ms.

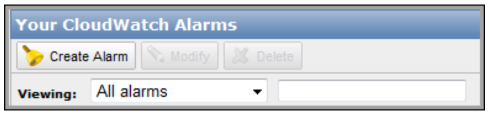
AWS Management Console

The **Create Alarm Wizard** steps you through the process of creating an alarm.

To open the Create Alarm Wizard

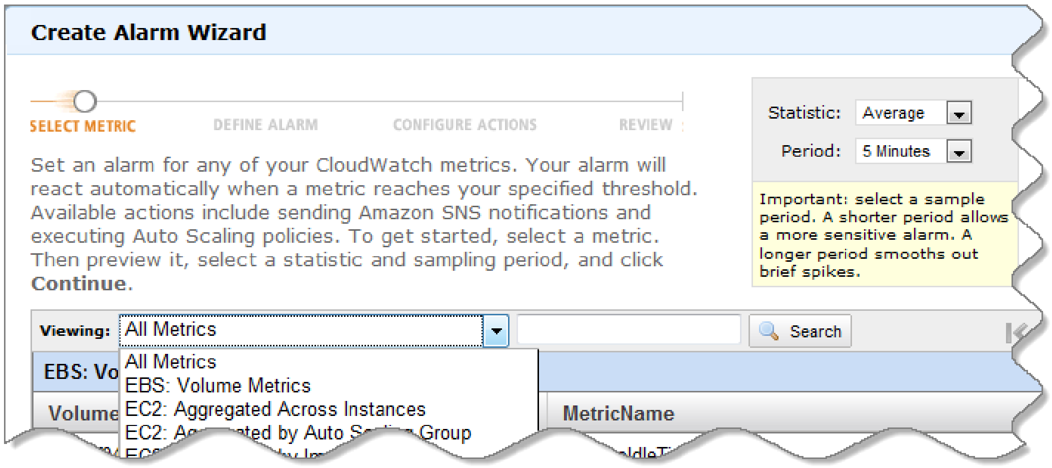
1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Alarms**.

   The **Your CloudWatch Alarms** page opens.



3. Click **Create Alarm**.

   The **SELECT METRIC** page of the **Create Alarm Wizard** opens.



Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

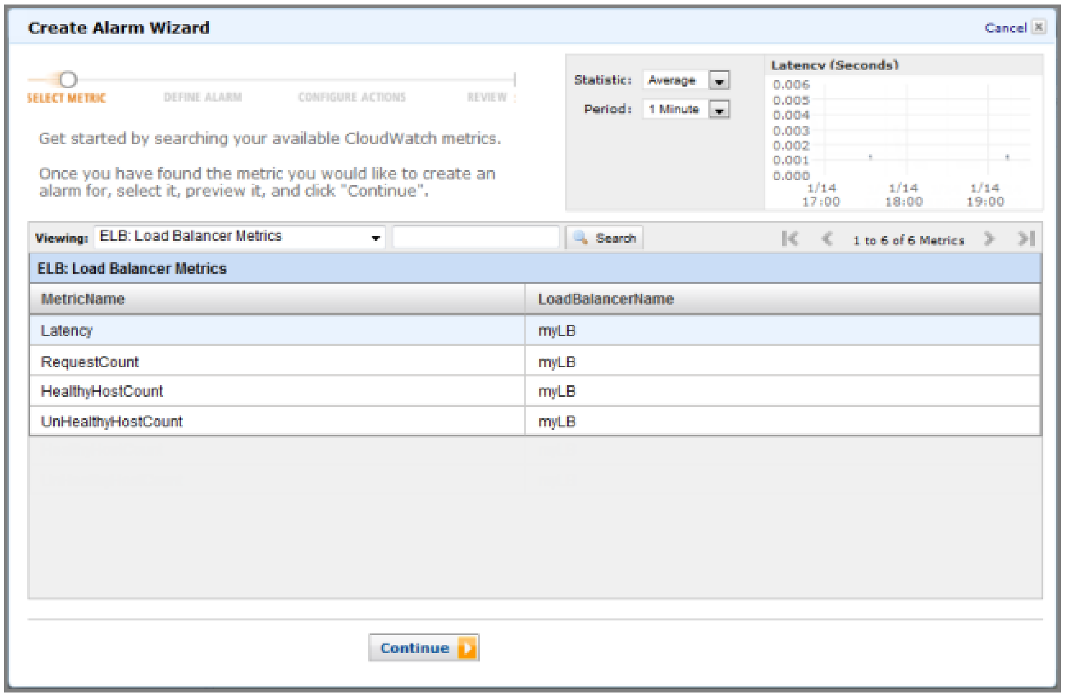
Important: select a sample period. A shorter period allows a more sensitive alarm. A longer period smooths out brief spikes.

To select a metric for your alarm

1. Select **ELB: Load Balancer Metrics** from the **Viewing** drop-down list.

   The metrics available for individual instances appear in the **Metrics** pane.
2. Select a row that contains **Latency** for a specific load balancer.

   A graph showing average `Latency` for a single load balancer appears next to the **Statistic** and **Period** drop-down lists.



Get started by searching your available CloudWatch metrics.

Once you have found the metric you would like to create an alarm for, select it, preview it, and click "Continue".

Statistic: Average
Period: 1 Minute

Viewing: ELB: Load Balancer Metrics
1 to 6 of 6 Metrics

ELB: Load Balancer Metrics

MetricName	LoadBalancerName
Latency	myLB
RequestCount	myLB
HealthyHostCount	myLB
UnHealthyHostCount	myLB

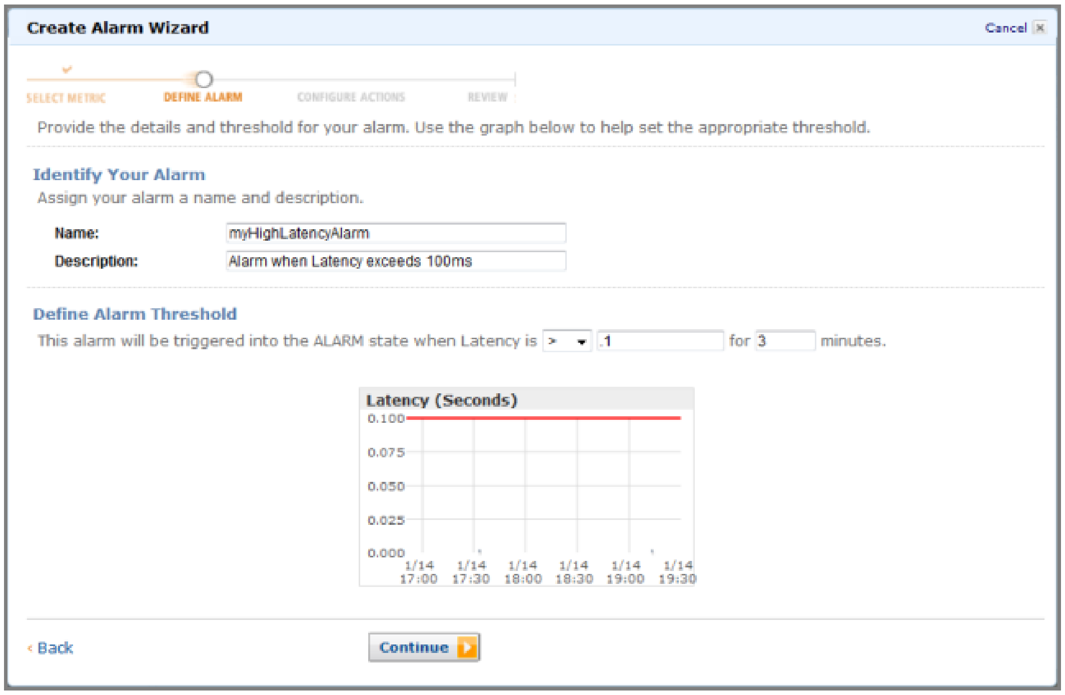
3. Select **Average** from the **Statistic** drop-down list.
4. Select **1 Minute** from the **Period** drop-down list.
5. Click **Continue**.

   The **DEFINE ALARM** page of the **Create Alarm Wizard** opens.

To define the alarm name, description, and threshold

1. In the **Name** field, enter the name of the alarm, for example: **`myHighCpuAlarm`**.
2. In the **Description** field, enter a description of the alarm, for example: **`Alarm when Latency exceeds 100ms.`**
3. Select **>** in the **Define Alarm Threshold** drop-down list.
4. Enter **`0.1`** in the first **Define Alarm Threshold** field and **3** in the second field.

   A graphical representation of the threshold appears on the page.



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

Identify Your Alarm

Assign your alarm a name and description.

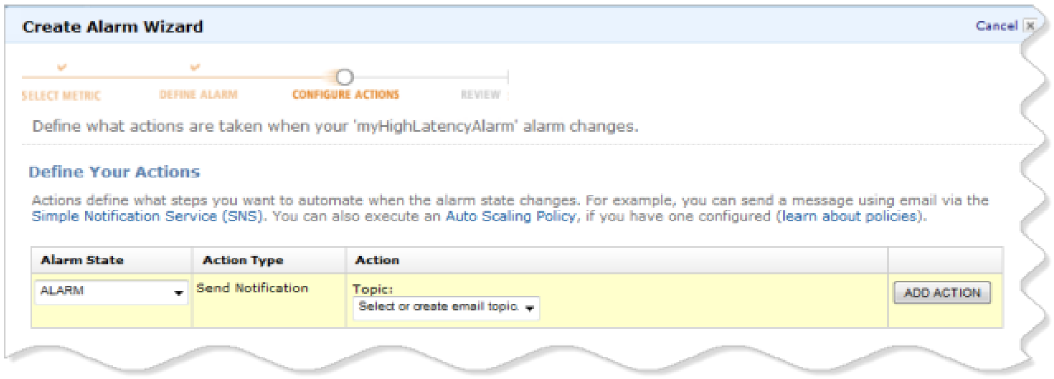
Name:	myHighLatencyAlarm
Description:	Alarm when Latency exceeds 100ms

Define Alarm Threshold

This alarm will be triggered into the ALARM state when Latency is > .1 for 3 minutes.

5. Click **Continue**.

   The **CONFIGURE ACTIONS** page of the **Create Alarm Wizard** opens.



Define what actions are taken when your 'myHighLatencyAlarm' alarm changes.

Define Your Actions

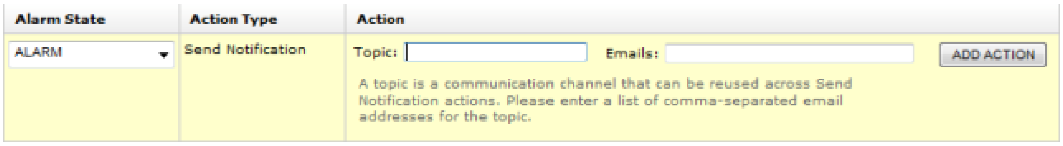
Actions define what steps you want to automate when the alarm state changes. For example, you can send a message using email via the Simple Notification Service (SNS). You can also execute an Auto Scaling Policy, if you have one configured (learn about policies).

Alarm State	Action Type	Action	
ALARM	Send Notification	Topic: Select or create email topic.	ADD ACTION

To configure an email action for an alarm

1. Select **ALARM** from the **Alarm State** drop-down list.
2. Select **Create Email Topic...** from the **Topic** drop-down list.

   Two new fields named **Topic** and **Emails** replace the **Topic** drop-down list.

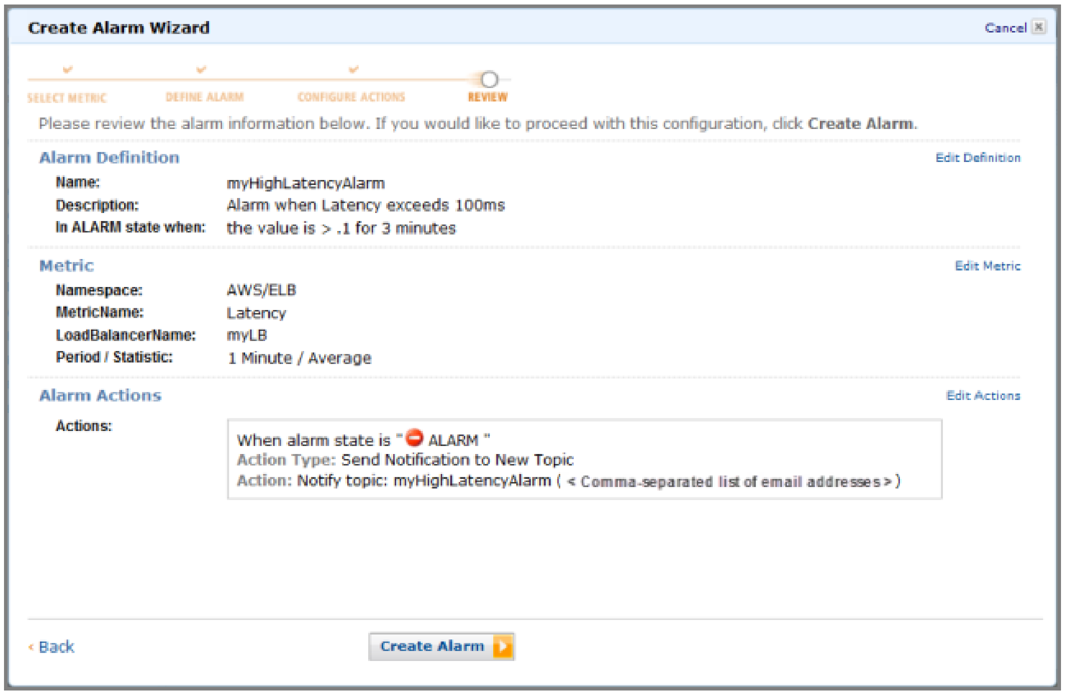


Alarm State	Action Type	Action	
ALARM	Send Notification	Topic: Emails: A topic is a communication channel that can be reused across Send Notification actions. Please enter a list of comma-separated email addresses for the topic.	ADD ACTION

3. In the **Topic** field, enter a descriptive name for the Amazon SNS topic, for example: **`myHighCpuAlarm`**.
4. In the **Emails** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state.
5. Click **ADD ACTION**.

   The action is saved and the **ADD ACTION** button becomes a **REMOVE** button.
6. Click **Continue**.

   The **REVIEW** window of the **Create Alarm Wizard** opens.



Please review the alarm information below. If you would like to proceed with this configuration, click **Create Alarm**.

Alarm Definition

Name:	myHighLatencyAlarm
Description:	Alarm when Latency exceeds 100ms
In ALARM state when:	the value is > .1 for 3 minutes

Metric

Namespace:	AWS/ELB
MetricName:	Latency
LoadBalancerName:	myLB
Period / Statistic:	1 Minute / Average

Alarm Actions

Actions: When alarm state is " ALARM "
Action Type: Send Notification to New Topic
Action: Notify topic: myHighLatencyAlarm (< Comma-separated list of email addresses >)

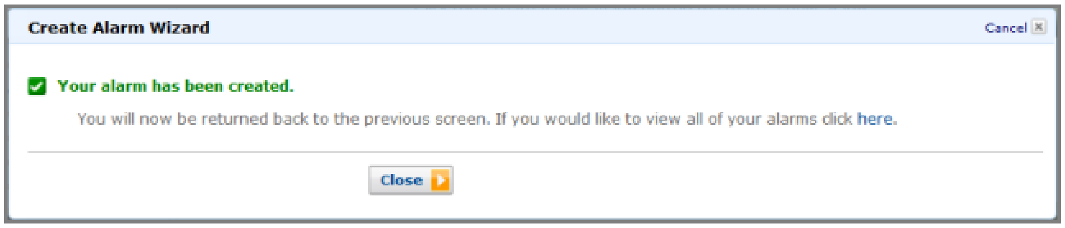
Now that you have defined the alarm and configured the alarm's actions, you are ready to review the settings and create the alarm by following the steps in the next procedure.

To review the alarm settings and create the alarm

1. Review the alarm settings presented in the **REVIEW** page of the **Create Alarm Wizard**.

   You can make changes to the settings using the **Edit Definition**, **Edit Metric**, or **Edit Actions** links.
2. Click **Create Alarm** to complete the alarm creation process.

   A confirmation dialog box opens.



Your alarm has been created.

You will now be returned back to the previous screen. If you would like to view all of your alarms click here.

3. Click **Close**.

Your alarm is created.

Command Line Tools

To send an Amazon SNS email message when LoadBalancer Latency Exceeds 100 milliseconds

1. Create an Amazon SNS topic. See instructions for creating an Amazon SNS topic in Set Up Amazon SNS (p. 67)
2. Create the alarm.

```
Prompt>mon-put-metric-alarm  --alarm-name lb-mon --alarm-description "Alarm
 when Latency exceeds 100ms" --metric-name Latency --namespace AWS/ELB --
statistic Average  --period 60 --threshold 100 --comparison-operator Great
erThanThreshold  --dimensions LoadBalancerName=my-server  --evaluation-
periods 3 --alarm-actions arn:aws:sns:us-east-1:1234567890:my-topic --unit
 Milliseconds
```

   Amazon CloudWatch returns the following:

```
OK-Created Alarm
```

3. Test the alarm.

   - Force an alarm state change to ALARM:

```
Prompt>mon-set-alarm-state  --state OK
Prompt>mon-set-alarm-state  --state ALARM
```

     Amazon CloudWatch returns the following:

```
OK-Set alarm state value
```

   - Check that an email has been received.

Send Email Based on Storage Throughput Alarm

Topics

- AWS Management Console (p. 80)
- Command Line Tools (p. 85)

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

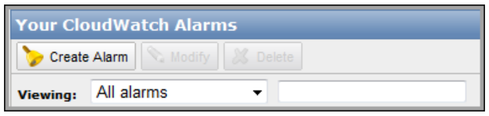
AWS Management Console

The **Create Alarm Wizard** steps you through the process of creating an alarm.

To open the Create Alarm Wizard

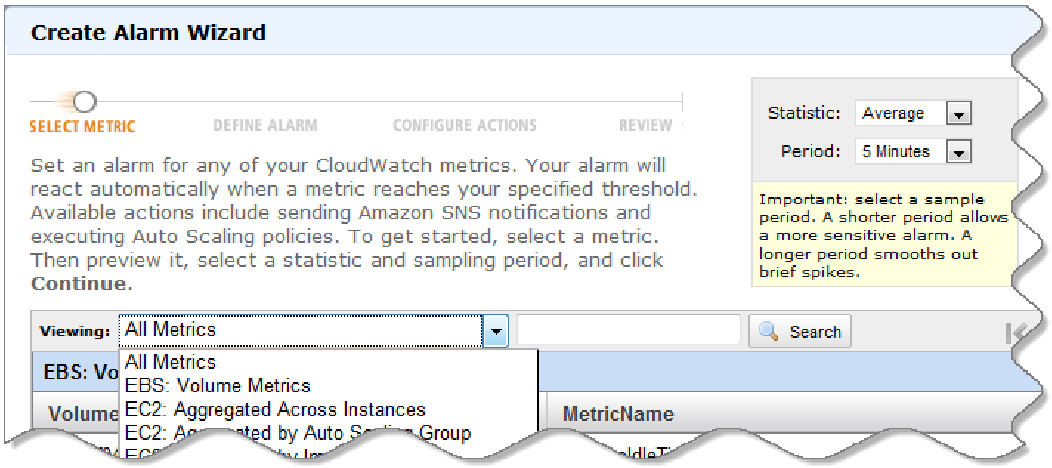
1. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Alarms**.

   The **Your CloudWatch Alarms** page opens.



3. Click **Create Alarm**.

   The **SELECT METRIC** page of the **Create Alarm Wizard** opens.


Create Alarm Wizard

Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

Statistic: Average
Period: 5 Minutes

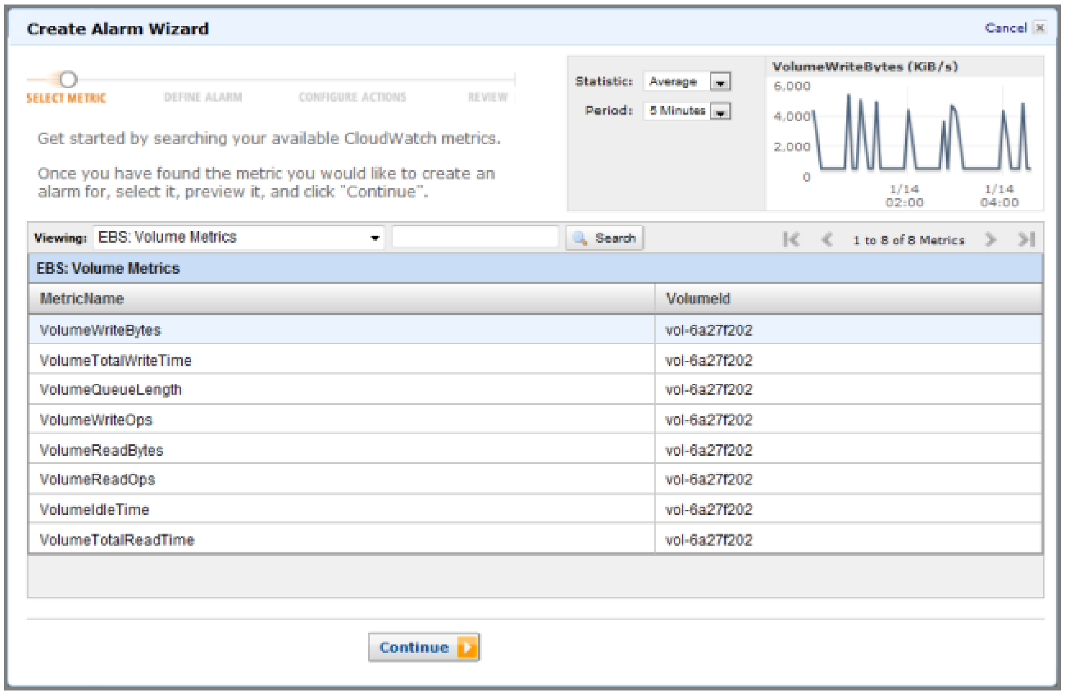
Important: select a sample period. A shorter period allows a more sensitive alarm. A longer period smooths out brief spikes.

Viewing: All Metrics

To select a metric for your alarm

1. In the **SELECT METRIC** page of the **Create Alarm Wizard**, select **EBS: Volume Metrics** from the **Viewing** drop-down list.
2. Select a row that contains **VolumeWriteBytes** for a specific VolumeId.

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



Get started by searching your available CloudWatch metrics.

Once you have found the metric you would like to create an alarm for, select it, preview it, and click "Continue".

Viewing: EBS: Volume Metrics

EBS: Volume Metrics

MetricName	VolumeId
VolumeWriteBytes	vol-6a27f202
VolumeTotalWriteTime	vol-6a27f202
VolumeQueueLength	vol-6a27f202
VolumeWriteOps	vol-6a27f202
VolumeReadBytes	vol-6a27f202
VolumeReadOps	vol-6a27f202
VolumeIdleTime	vol-6a27f202
VolumeTotalReadTime	vol-6a27f202

3. Select **Average** from the **Statistic** drop-down list.

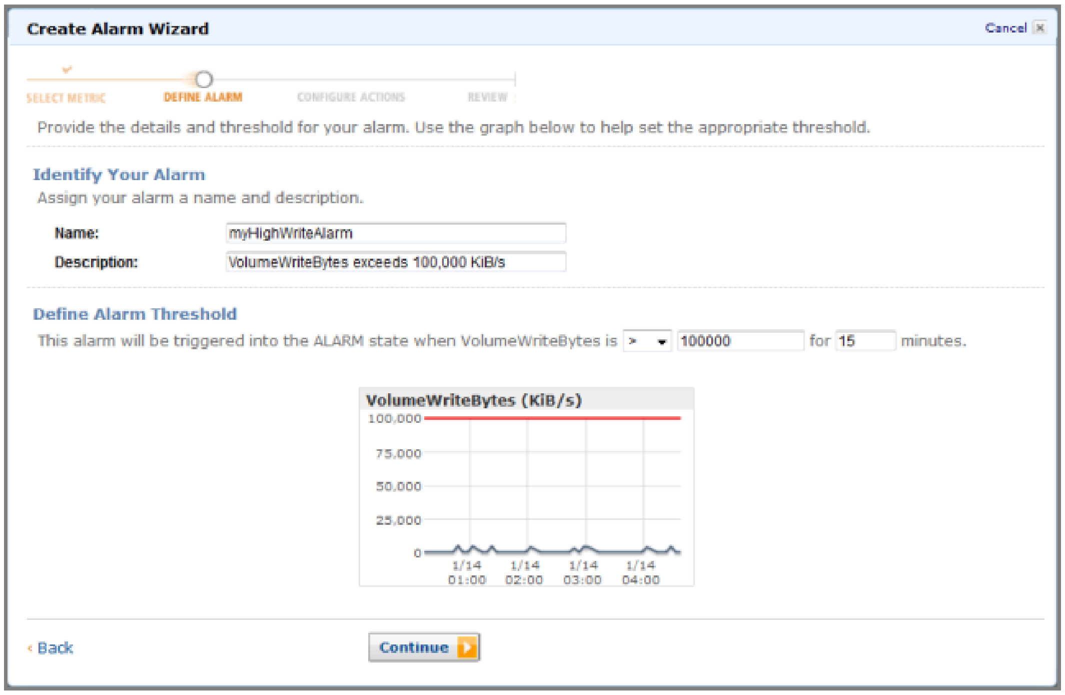
   The metrics available for individual volumes appear in the **Metrics** pane.
4. Select **5 Minutes** from the **Period** drop-down list.
5. Click **Continue**.

   The **DEFINE ALARM** page of the **Create Alarm Wizard** opens.

To define the alarm name, description, and threshold

1. On the **DEFINE ALARM** page of the **Create Alarm Wizard**, in the **Name** field, enter the name of the alarm, for example: **`myHighWriteAlarm`**.
2. In the **Description** field, enter a description of the alarm, for example: **`VolumeWriteBytes exceeds 100,000 KiB/s.`**
3. Select **>** in the **Define Alarm Threshold** drop-down list.
4. Enter a threshold value in the first **Define Alarm Threshold** field and a duration value in the second field, for example: **`100000`** for the threshold and **`15`** for the duration.

   A graphical representation of the threshold appears on the page.



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

Identify Your Alarm

Assign your alarm a name and description.

Name: myHighWriteAlarm

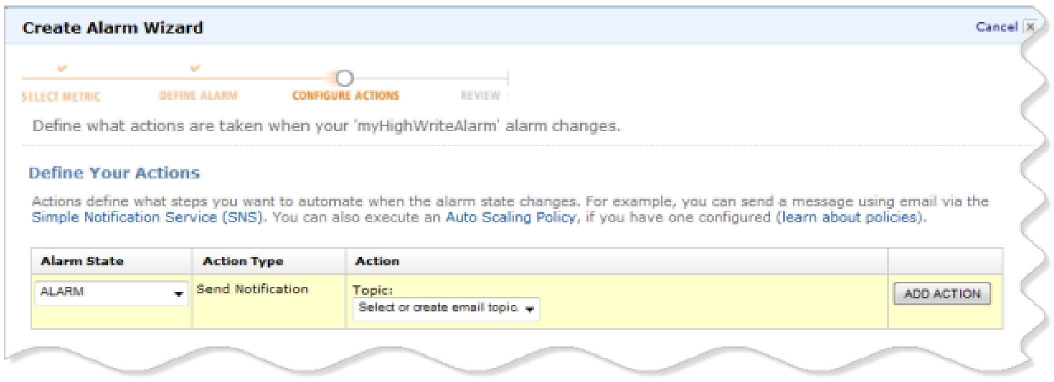
Description: VolumeWriteBytes exceeds 100,000 KiB/s

Define Alarm Threshold

This alarm will be triggered into the ALARM state when VolumeWriteBytes is > 100000 for 15 minutes.

5. Click **Continue**.

   The **CONFIGURE ACTIONS** page of the **Create Alarm Wizard** opens.



Define what actions are taken when your 'myHighWriteAlarm' alarm changes.

Define Your Actions

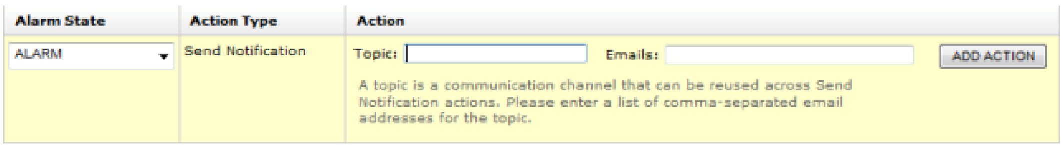
Actions define what steps you want to automate when the alarm state changes. For example, you can send a message using email via the Simple Notification Service (SNS). You can also execute an Auto Scaling Policy, if you have one configured (learn about policies).

Alarm State	Action Type	Action
ALARM	Send Notification	Topic: Select or create email topic.

To configure an email action for an alarm

1. On the **CONFIGURE ACTIONS** page of the **Create Alarm Wizard**, select **ALARM** from the **Alarm State** drop-down list.
2. Select **Create Email Topic...** from the **Topic** drop-down list.

   Two new fields named **Topic** and **Emails** replace the **Topic** drop-down list.

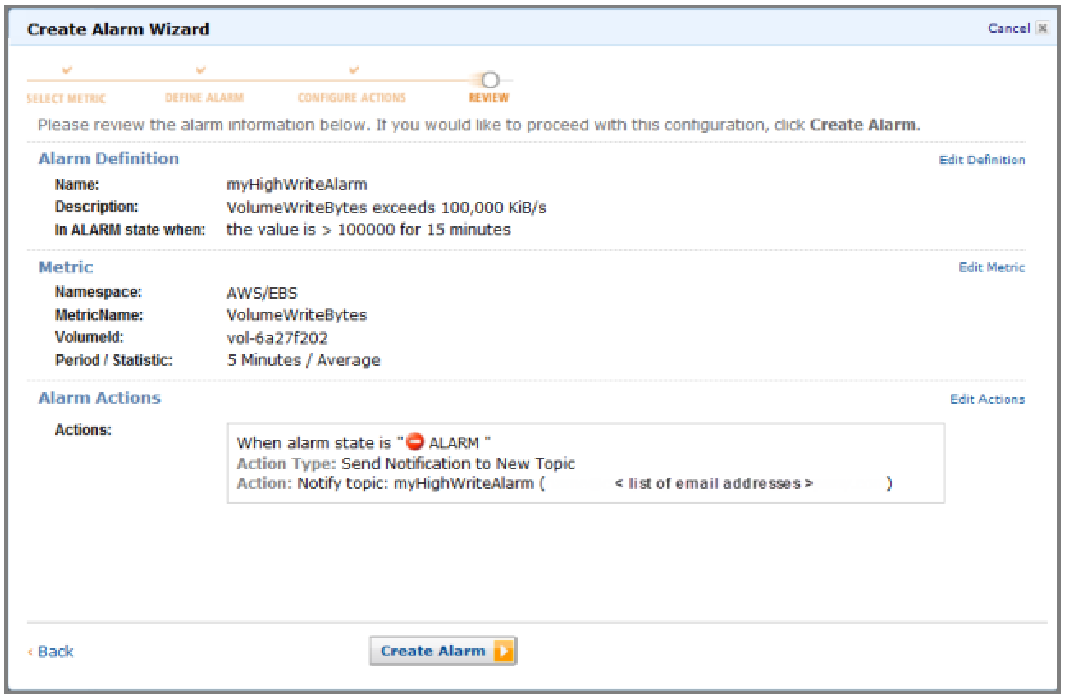


Alarm State	Action Type	Action
ALARM	Send Notification	Topic: Emails: A topic is a communication channel that can be reused across Send Notification actions. Please enter a list of comma-separated email addresses for the topic.

3. In the **Topic** field, enter a descriptive name for the Amazon SNS topic, for example: **`myHighWriteAlarm`**.
4. In the **Emails** field, enter a comma-separated list of email addresses to be notified when the alarm changes to the `ALARM` state.
5. Click **Add Action**.

   The action is saved and the **Add Action** button becomes a **Remove** button.
6. Click **Continue**.

   The **REVIEW** page of the **Create Alarm Wizard** opens.



Please review the alarm information below. If you would like to proceed with this configuration, click Create Alarm.

Alarm Definition — Edit Definition

Name:	myHighWriteAlarm
Description:	VolumeWriteBytes exceeds 100,000 KiB/s
In ALARM state when:	the value is > 100000 for 15 minutes

Metric — Edit Metric

Namespace:	AWS/EBS
MetricName:	VolumeWriteBytes
VolumeId:	vol-6a27f202
Period / Statistic:	5 Minutes / Average

Alarm Actions — Edit Actions

Actions:
When alarm state is " ALARM "
Action Type: Send Notification to New Topic
Action: Notify topic: myHighWriteAlarm (< list of email addresses >)

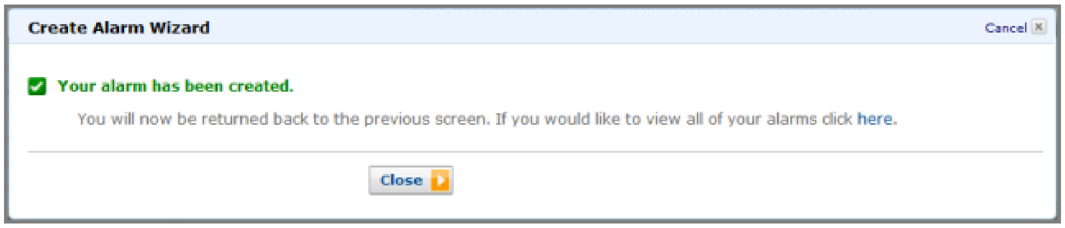
Now that you have defined the alarm and configured the alarm's actions, you are ready to review the settings and create the alarm.

To review the alarm settings and create the alarm

1. Review the alarm settings presented in the **REVIEW** window of the **Create Alarm Wizard**.

   You can make changes to the settings using the **Edit Definition**, **Edit Metric**, or **Edit Actions** links.
2. Click **Create Alarm** to complete the alarm creation process.

   A confirmation dialog box opens.



Your alarm has been created.

You will now be returned back to the previous screen. If you would like to view all of your alarms click here.

3. Click **Close**.

   Your alarm is created.

Command Line Tools

To send an Amazon SNS email message when EBS exceeds 100 MB throughput

1. Create an Amazon SNS topic. See instructions for creating an Amazon SNS topic in Set Up Amazon SNS (p. 67).
2. Create the alarm.

```
Prompt>mon-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 VolumeId=my-volume-
id --evaluation-periods 3 --alarm-actions arn:aws:sns:us-east-1:1234567890:my-
alarm-topic --insufficient-data-actions arn:aws:sns:us-east-1:1234567890:my-
insufficient-data-topic
```

   Amazon CloudWatch returns the following:

```
OK-Created Alarm
```

3. Test the alarm.

   - Force an alarm state change to ALARM.

```
Prompt>mon-set-alarm-state  --state OK
Prompt>mon-set-alarm-state  --state ALARM
Prompt>mon-set-alarm-state  --state INSUFFICIENT_DATA
```

   - Check that two emails have been received.

Monitor Your Estimated Charges Using Amazon CloudWatch

Topics

You can monitor your estimated Amazon Web Services (AWS) charges using Amazon CloudWatch. When you enable the monitoring of estimated charges for your AWS account, the estimated charges are calculated and sent several times daily to Amazon CloudWatch as metric data that is stored for 14 days. This data includes the estimated charges for every service in AWS that you use, as well the estimated overall total of your AWS charges. You can choose to receive alerts by email when charges have exceeded a certain threshold. These alerts are triggered by Amazon CloudWatch and are sent using Amazon Simple Notification Service (Amazon SNS) notification.

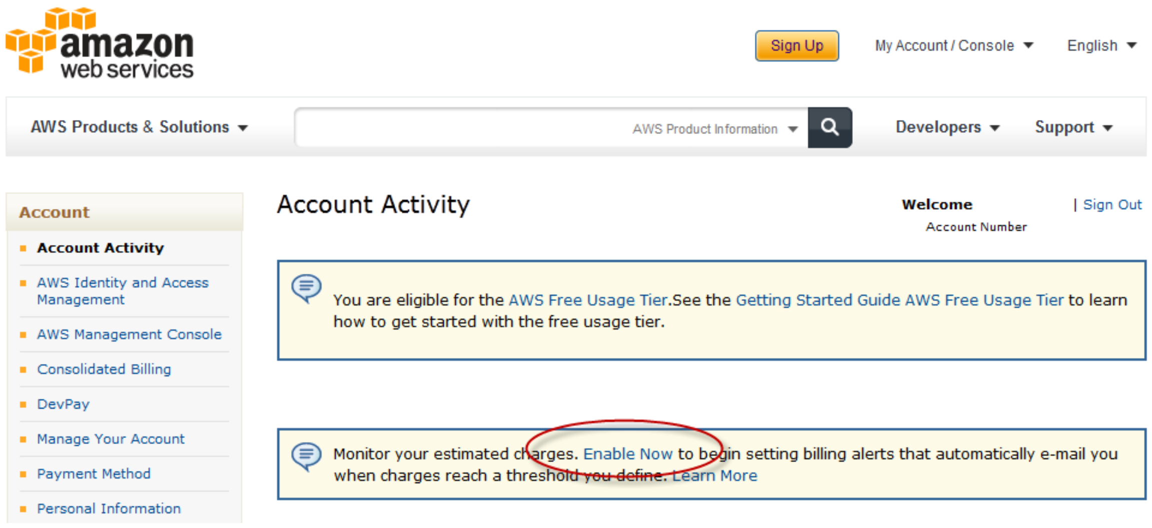
The metrics are provided free of charge, and you get 10 Amazon CloudWatch alarms and 1,000 Amazon SNS email notifications per customer per month for free. Any additional alarms or email notifications are priced at standard AWS rates. For more information, see Amazon CloudWatch Pricing, and Amazon SNS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

Monitoring Your Estimated Charges Using Amazon CloudWatch

Before you can create an alarm on your estimated charges, you must enable monitoring. When you enable the monitoring of your estimated charges, this creates metric data that you can use to create a billing alarm. It takes about 15 minutes before you can view billing data and create alarms using the Amazon CloudWatch console or command line interface (CLI). After you enable billing metrics you cannot disable the collection of data, but you can delete any alarms you have created.

To enable the monitoring of estimated charges

1. Go to the Amazon Web Services website at http://aws.amazon.com.
2. Click **My Account/Console**, and then click **Account Activity**.
3. In the spaces provided, enter your user name and password, and then click **Sign in using our secure server**.
4. Under **Account Activity**, in **Monitor your estimated charges** box, click **Enable Now**.



Account Activity

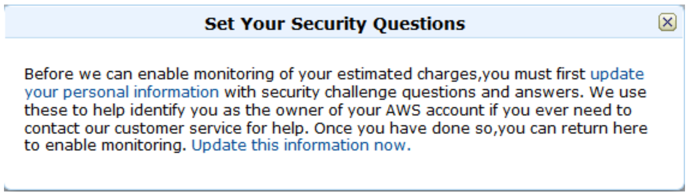
You are eligible for the AWS Free Usage Tier.See the Getting Started Guide AWS Free Usage Tier to learn how to get started with the free usage tier.

Monitor your estimated charges. Enable Now to begin setting billing alerts that automatically e-mail you when charges reach a threshold you define. Learn More

Note

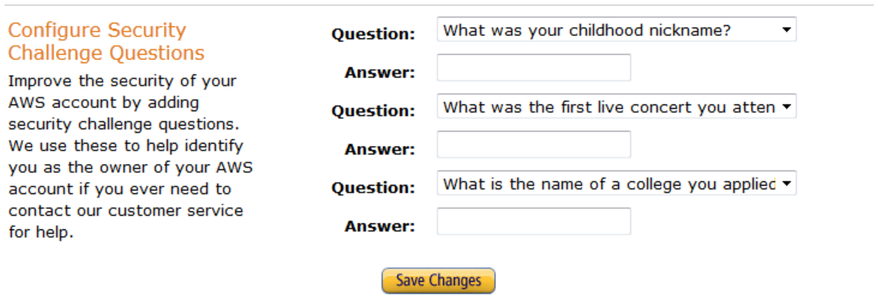
If you haven't already provided security challenge information, you must select three questions and provide answers for each. This information will be used to verify your identity if you need to contact customer service for help, and must be provided before you can enable monitoring of your charges.

a. On the **Set Your Security Questions** dialog box, click **update your personal information**.


Set Your Security Questions

Before we can enable monitoring of your estimated charges,you must first update your personal information with security challenge questions and answers. We use these to help identify you as the owner of your AWS account if you ever need to contact our customer service for help. Once you have done so,you can return here to enable monitoring. Update this information now.

b. On the **Account Activity** page, under **Configure Security Challenge Questions**, choose three questions, provide answers for each, and then click **Save Changes**.

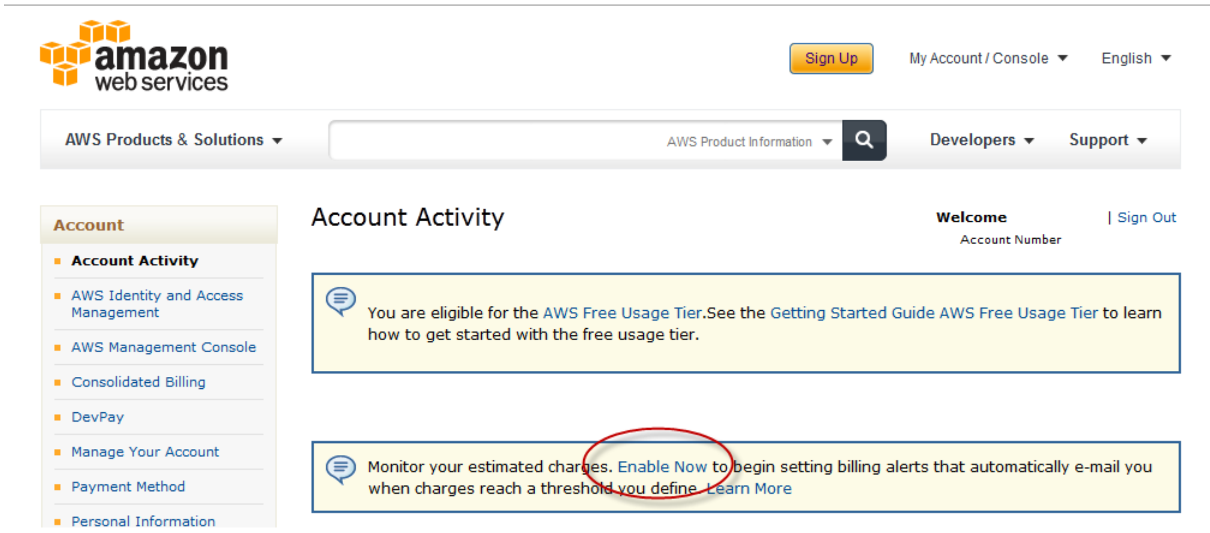

Configure Security Challenge Questions

Improve the security of your AWS account by adding security challenge questions. We use these to help identify you as the owner of your AWS account if you ever need to contact our customer service for help.

Question:	What was your childhood nickname?
Answer:	
Question:	What was the first live concert you atten
Answer:	
Question:	What is the name of a college you appliec
Answer:	

Save Changes

c. On the **Account Activity** page, in the **Monitor your estimated charges** box, click **Enable Now**.



Account Activity

You are eligible for the AWS Free Usage Tier.See the Getting Started Guide AWS Free Usage Tier to learn how to get started with the free usage tier.

Monitor your estimated charges. Enable Now to begin setting billing alerts that automatically e-mail you when charges reach a threshold you define. Learn More

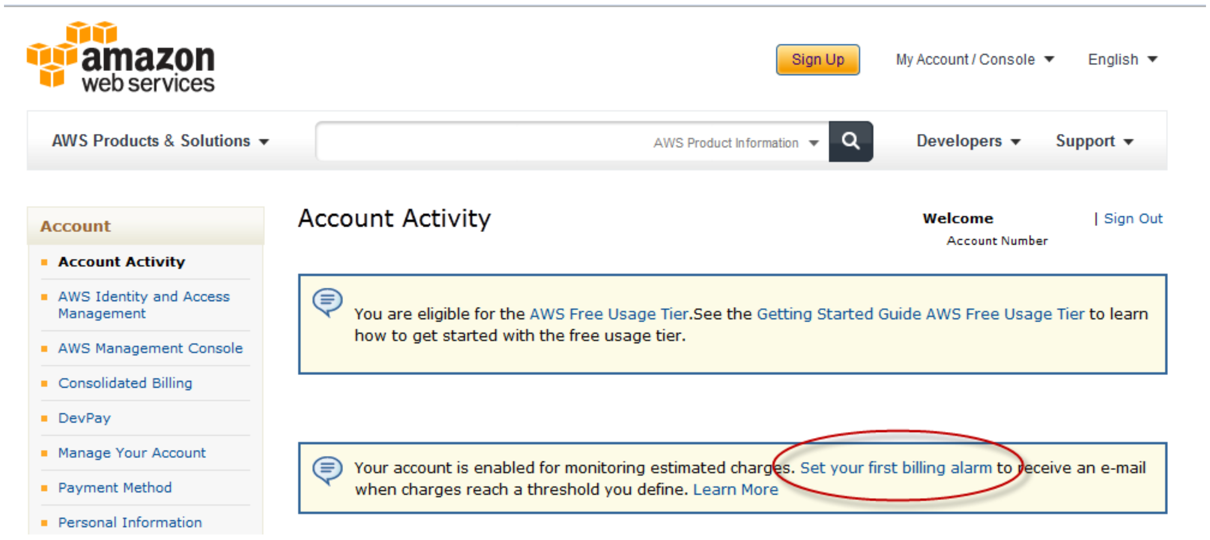
After you complete these steps, your account will be enabled. You will see a confirmation dialog box that reminds you it may take up to 15 minutes for billing data to be visible in Amazon CloudWatch. However, you can create an alarm immediately on your total AWS charges.

Creating a Billing Alarm

You can create a billing alarm from the **Account Activity** page or using the Amazon CloudWatch console. The following example creates an alarm that will send an email message when your estimated charges for AWS exceed $50. When you enable the monitoring of your estimated charges for the first time, it takes about 15 minutes before you can view billing data and set billing alarms using the Amazon CloudWatch console, or command line interface (CLI).

To create a billing alarm

1. Do one of the following:
   - On the **Account Activity** page, after you have enabled monitoring, click **Set your first billing alarm**.

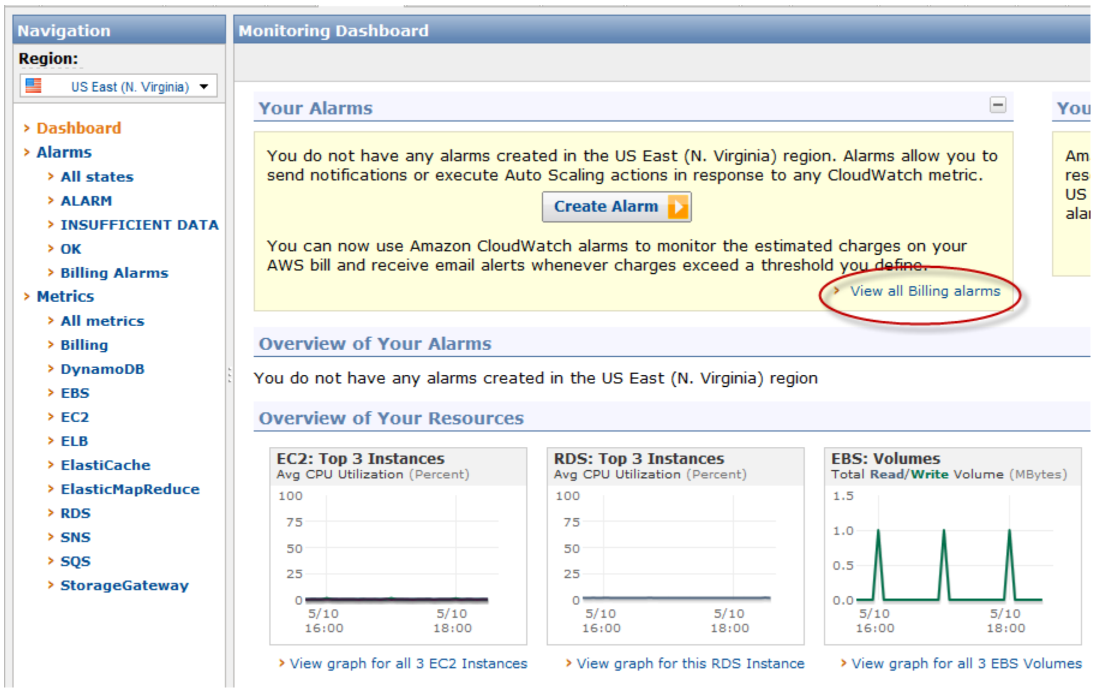


Account Activity

You are eligible for the AWS Free Usage Tier.See the Getting Started Guide AWS Free Usage Tier to learn how to get started with the free usage tier.

Your account is enabled for monitoring estimated charges. Set your first billing alarm to receive an e-mail when charges reach a threshold you define. Learn More

   - In the Amazon CloudWatch console, in the **Navigation** pane, click **Dashboard**. In the **Monitoring Dashboard** pane, under **Your Alarms**, click **View all Billing alarms**, and then click **Create Alarm**.



Your Alarms

You do not have any alarms created in the US East (N. Virginia) region. Alarms allow you to send notifications or execute Auto Scaling actions in response to any CloudWatch metric.

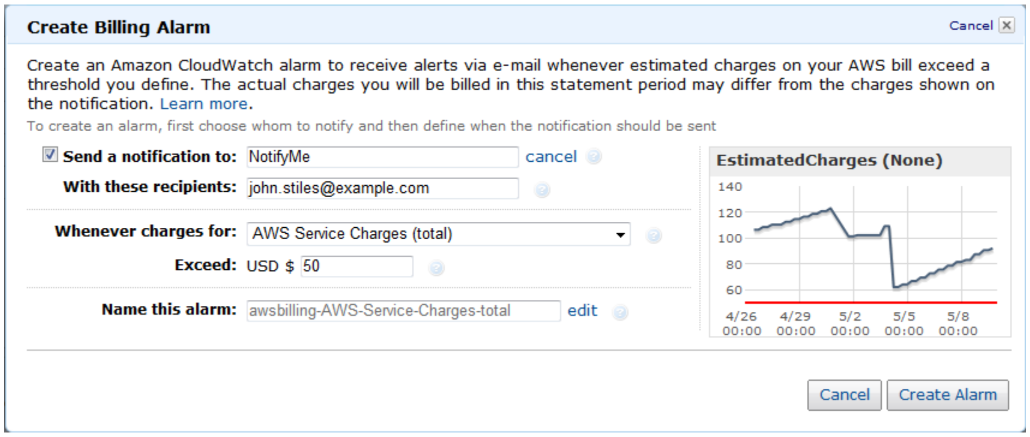
You can now use Amazon CloudWatch alarms to monitor the estimated charges on your AWS bill and receive email alerts whenever charges exceed a threshold you define.

Overview of Your Alarms

You do not have any alarms created in the US East (N. Virginia) region

Overview of Your Resources

2. In the **Create Billing Alarm** dialog box, in the **With these recipients** box, type your email address (e.g., john.stiles@example.com).


Create Billing Alarm

Create an Amazon CloudWatch alarm to receive alerts via e-mail whenever estimated charges on your AWS bill exceed a threshold you define. The actual charges you will be billed in this statement period may differ from the charges shown on the notification. Learn more.

To create an alarm, first choose whom to notify and then define when the notification should be sent

Send a notification to: NotifyMe
With these recipients: john.stiles@example.com
Whenever charges for: AWS Service Charges (total)
Exceed: USD $ 50
Name this alarm: awsbilling-AWS-Service-Charges-total

3. In the **Whenever charges for** drop-down menu, select the billing metric you want to be notified about (e.g., AWS Service Charges (total)), and then in the **Exceeds** box, set the monetary amount (e.g., 50) that must be exceeded to trigger the alarm and send an email.

   Note

   In the **Estimated Monthly Charges** thumbnail graph, you can see an estimate of your charges that you can use to set an appropriate threshold for the alarm.
4. In the **Name this alarm** box, type a friendly name for the alarm (e.g., AWSTotalCharges), and then click **Create Alarm**.

   Important

   If you added an email address to the list of recipients or created a new topic, Amazon SNS will send a subscription confirmation email to each new address shortly after you create an alarm. Remember to click the link contained in that message, which confirms your subscription. Alert notifications are only sent to confirmed addresses.
5. You can view any alarms you have created. In the Amazon CloudWatch console, in the **Navigation** pane, click **Billing Alarms**.

Creating a Billing Alarm Using the Create Alarm Wizard or CLI

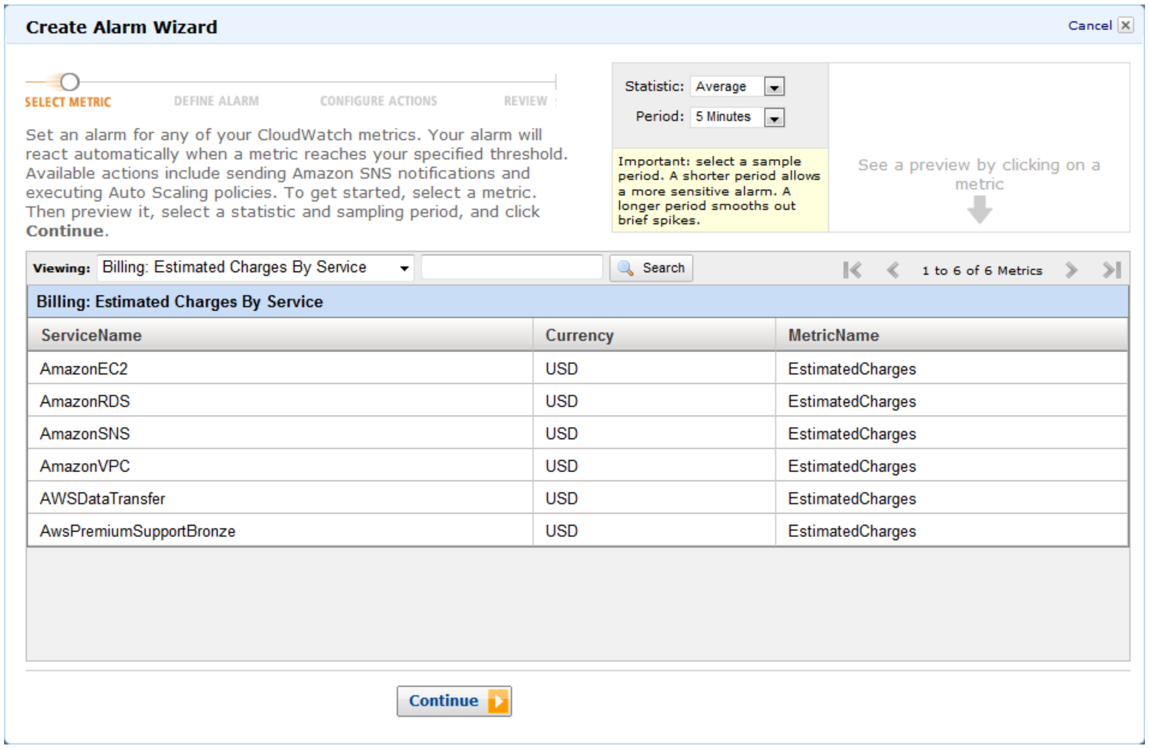
You can use either the **Create Alarm Wizard** in the Amazon CloudWatch console or the command line interface (CLI) to create a billing alarm for any service in AWS that you're using. You can apply more than one action to an alarm, or you can set the alarm to be triggered so that you receive an email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

To create a billing alarm using the Create Alarm Wizard

In this example, you can graph your charges for Amazon EC2 over the last 14 days and then set an alarm that sends email as soon as your charges exceed $50.

1. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Alarms**.
3. At the top of the **Your CloudWatch Alarms** pane, click **Create Alarm**.

4. In the **Create Alarm Wizard**, on the **SELECT METRIC** page, in the **Viewing** pop-up menu, select **Billing: Estimated Charges By Service** .



Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

Statistic: Average
Period: 5 Minutes

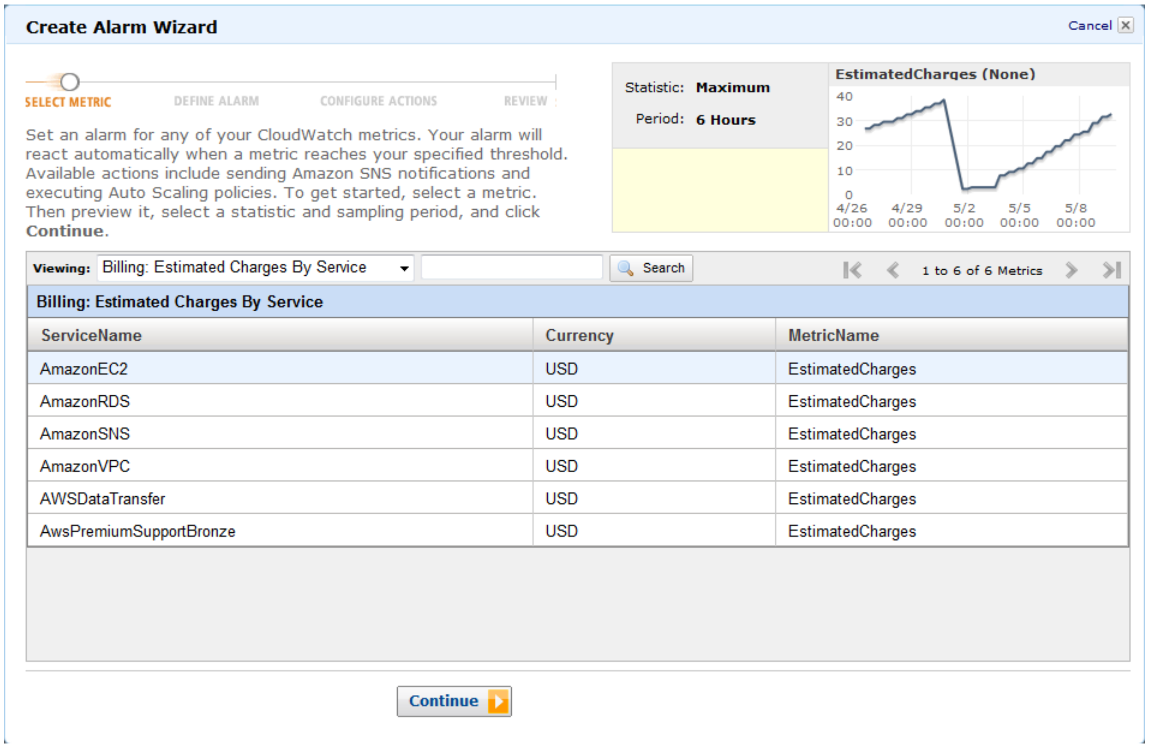
Important: select a sample period. A shorter period allows a more sensitive alarm. A longer period smooths out brief spikes.

Viewing: Billing: Estimated Charges By Service

Billing: Estimated Charges By Service

ServiceName	Currency	MetricName
AmazonEC2	USD	EstimatedCharges
AmazonRDS	USD	EstimatedCharges
AmazonSNS	USD	EstimatedCharges
AmazonVPC	USD	EstimatedCharges
AWSDataTransfer	USD	EstimatedCharges
AwsPremiumSupportBronze	USD	EstimatedCharges

5. In the **Billing: Estimated Charges By Service** list, select a service (e.g., AmazonEC2) to view its billing data in a graph in the upper-right corner of the page.



Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

Statistic: **Maximum**
Period: **6 Hours**

Viewing: Billing: Estimated Charges By Service

Billing: Estimated Charges By Service

ServiceName	Currency	MetricName
AmazonEC2	USD	EstimatedCharges
AmazonRDS	USD	EstimatedCharges
AmazonSNS	USD	EstimatedCharges
AmazonVPC	USD	EstimatedCharges
AWSDataTransfer	USD	EstimatedCharges
AwsPremiumSupportBronze	USD	EstimatedCharges

Important

When setting alarms on billing metrics, the **Create Alarm Wizard** automatically selects the statistic (such as Maximum) and period (such as 6 Hours) for a billing alarm. There is no need to change these selections.

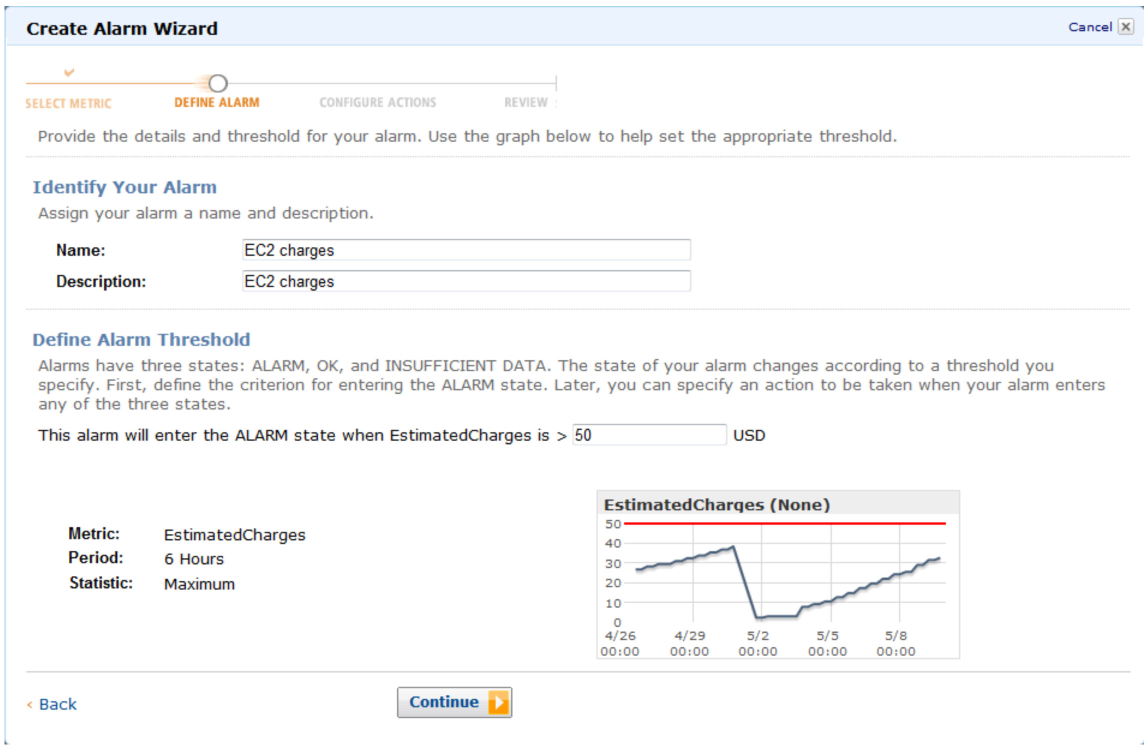
Note

There are several different ways you can view billing data for your account - **Billing: Estimated Charges Total**, **Billing: Estimated Charges By Service**, or **Billing: Estimated Charges By Linked Account**. For consolidated billing accounts, billing data for each linked account can be found by logging in as the paying account. You can view billing data for total estimated charges and estimated charges by service for each linked account as well as for the consolidated account.

6. Review the billing data and then click **Continue**.
7. On the **DEFINE ALARM** page, in the **Name** and **Description** boxes, enter a name for the billing alarm (e.g., EC2 charges) and a description (e.g., EC2).

   Note

   After you enter an alarm name, you cannot edit it later.



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

Identify Your Alarm

Assign your alarm a name and description.

Name: EC2 charges

Description: EC2 charges

Define Alarm Threshold

Alarms have three states: ALARM, OK, and INSUFFICIENT DATA. The state of your alarm changes according to a threshold you specify. First, define the criterion for entering the ALARM state. Later, you can specify an action to be taken when your alarm enters any of the three states.

This alarm will enter the ALARM state when EstimatedCharges is > 50 USD

Metric: EstimatedCharges

Period: 6 Hours

Statistic: Maximum

8. On the **CONFIGURE ACTIONS** page, in the **When Alarm state is** pop-up menu, select the alarm state (**ALARM**, **OK**, **INSUFFICIENT_DATA**) that you want to define. In most cases, you should choose the **ALARM** state that triggers when the threshold has been reached.

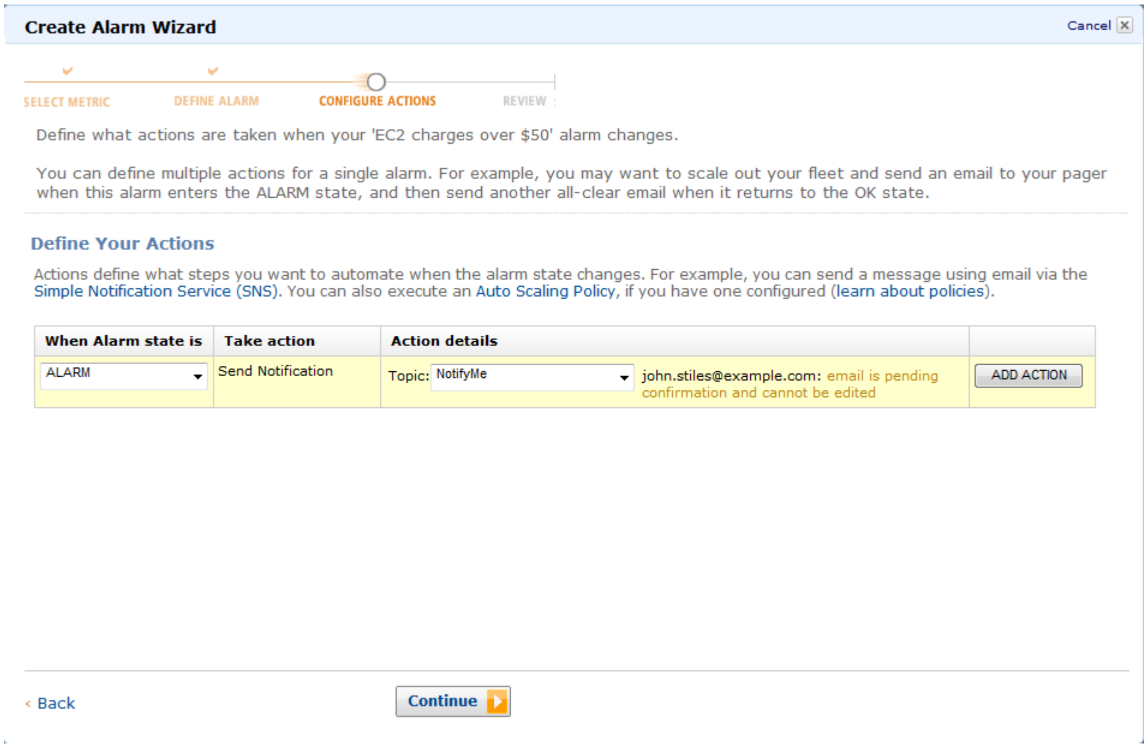
   Note

   There are three states that alarms can be in depending on the specified thresholds:

   - **ALARM:** Selected service has reached specified threshold.
   - **OK:** Selected service has not reached specified threshold.

- **INSUFFICIENT DATA:** No metrics are collected; often this service is not accruing charges. When using the recommended settings, this state is not common.

For the example in this procedure, the alarm is triggered when the estimated charges are greater than or equal to $50, as determined by the metric you select.



Define what actions are taken when your 'EC2 charges over $50' alarm changes.

You can define multiple actions for a single alarm. For example, you may want to scale out your fleet and send an email to your pager when this alarm enters the ALARM state, and then send another all-clear email when it returns to the OK state.

Define Your Actions

Actions define what steps you want to automate when the alarm state changes. For example, you can send a message using email via the Simple Notification Service (SNS). You can also execute an Auto Scaling Policy, if you have one configured (learn about policies).

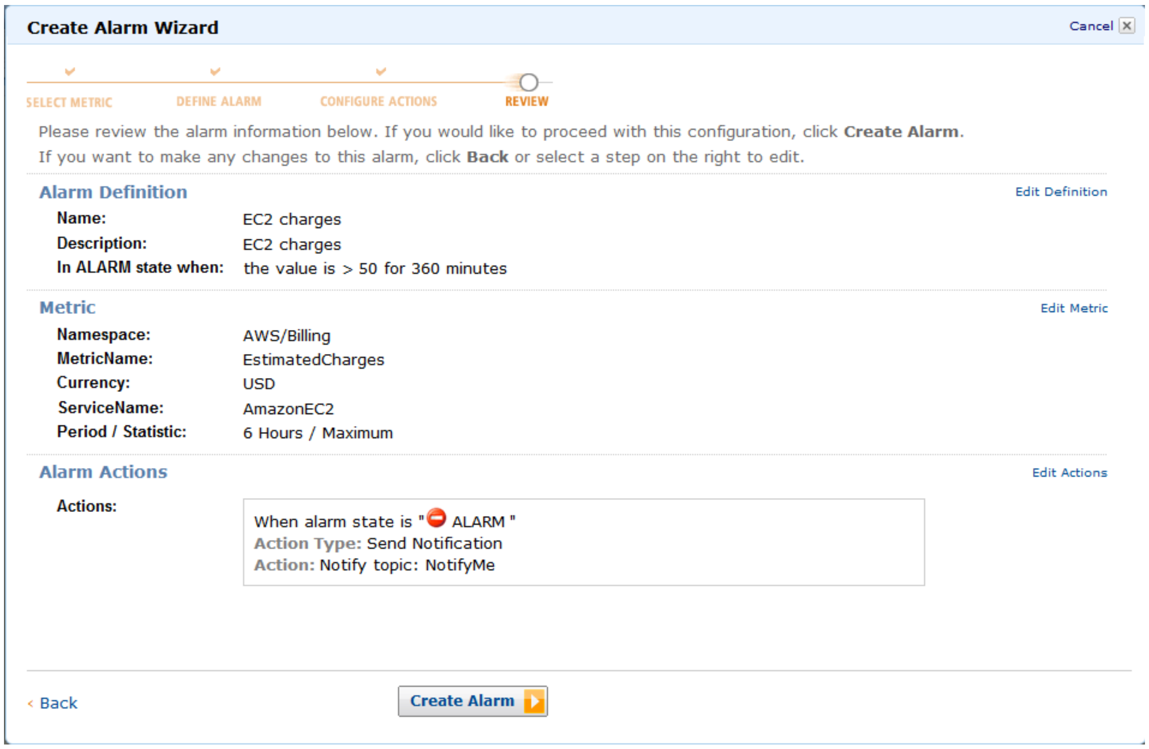
When Alarm state is	Take action	Action details	
ALARM	Send Notification	Topic: NotifyMe john.stiles@example.com: email is pending confirmation and cannot be edited	ADD ACTION

9. In the **Take action** pop-up menu, select **Send Notification**, and then in the **Action details** pop-up menu, select **Create New Email Topic**.

   Important

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

10. To define additional actions for the alarm, click **Add Action**. When you are through adding actions for the alarm, click **Continue**.
11. On the **REVIEW** page, review the alarm and edit it if needed. When you're satisfied with the alarm, click **Create Alarm**.



Please review the alarm information below. If you would like to proceed with this configuration, click **Create Alarm**.
If you want to make any changes to this alarm, click **Back** or select a step on the right to edit.

Alarm Definition — Edit Definition

Name:	EC2 charges
Description:	EC2 charges
In ALARM state when:	the value is > 50 for 360 minutes

Metric — Edit Metric

Namespace:	AWS/Billing
MetricName:	EstimatedCharges
Currency:	USD
ServiceName:	AmazonEC2
Period / Statistic:	6 Hours / Maximum

Alarm Actions — Edit Actions

Actions:
When alarm state is " ALARM "
Action Type: Send Notification
Action: Notify topic: NotifyMe

To create a billing alarm using the CLI

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

The example in the following procedure creates an alarm that will send an email message when your estimated monthly charges for Amazon EC2 exceed $50. For more information about the Amazon CloudWatch CLI, see Amazon CloudWatch Command Line Interface Reference (p. 98).

1. At a command prompt, type `mon-list-metrics --headers` to view the list of all available Amazon CloudWatch metrics for the services in AWS that you're using.
2. In the list of metrics, look in the **Namespace** column (second column), and review the billing metrics that have the AWS/Billing namespace. These are the billing metrics that you can use to create a billing alarm.
3. At the command prompt, enter the following command:

```
mon-put-metric-alarm ec2billing --comparison-operator
      GreaterThanOrEqualToThreshold --evaluation-periods 1 --metric-name Es
timatedCharges
      --namespace AWS/Billing --dimensions "Currency=USD" --period 21600 --
statistic Maximum --threshold 50 --actions-enabled true --alarm-actions
arn:aws:sns:us-east-1:111111111111:NotifyMe
```

   Where:

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

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

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

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

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

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

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

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

The **--actions-enabled** is whether the alarm should perform an action. Set it to true to perform the --alarm-actions (e.g., send an email), or set it to false to disregard the --alarm-actions.

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

Note

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

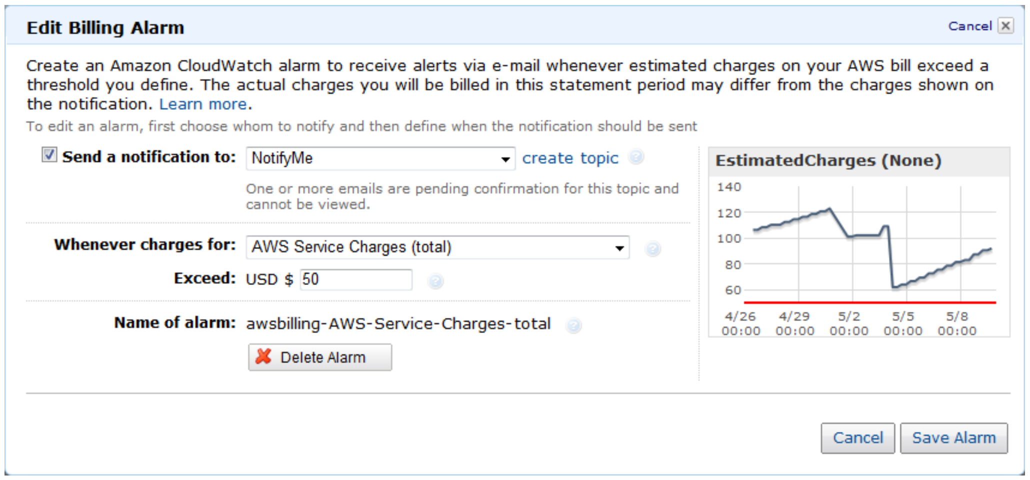
a. Open the Amazon SNS console at https://console.aws.amazon.com/sns/.
b. On the **Navigation** pane, under **My Topics**, select the topic you want the alarm to send mail to.
c. The ARN is located in the **Topic ARN** field on the **Topic Details** pane.

Editing a Billing Alarm

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

To edit a billing alarm using the Amazon CloudWatch console

1. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Billing Alarms**.
3. In the **Alarm details for AWS billing estimated charges** dialog box, click the alarm you want to change.
4. In the **Edit Billing Alarm** dialog box, make the necessary changes, and then click **Save Alarm**.


Edit Billing Alarm

Create an Amazon CloudWatch alarm to receive alerts via e-mail whenever estimated charges on your AWS bill exceed a threshold you define. The actual charges you will be billed in this statement period may differ from the charges shown on the notification. Learn more.

To edit an alarm, first choose whom to notify and then define when the notification should be sent

Send a notification to: NotifyMe

One or more emails are pending confirmation for this topic and cannot be viewed.

Whenever charges for: AWS Service Charges (total)

Exceed: USD $ 50

Name of alarm: awsbilling-AWS-Service-Charges-total

To edit a billing alarm using the CLI

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

Checking Alarm Status

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

To check alarm status using the Amazon CloudWatch console

1. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Billing Alarms**.

To check alarm status using the CLI

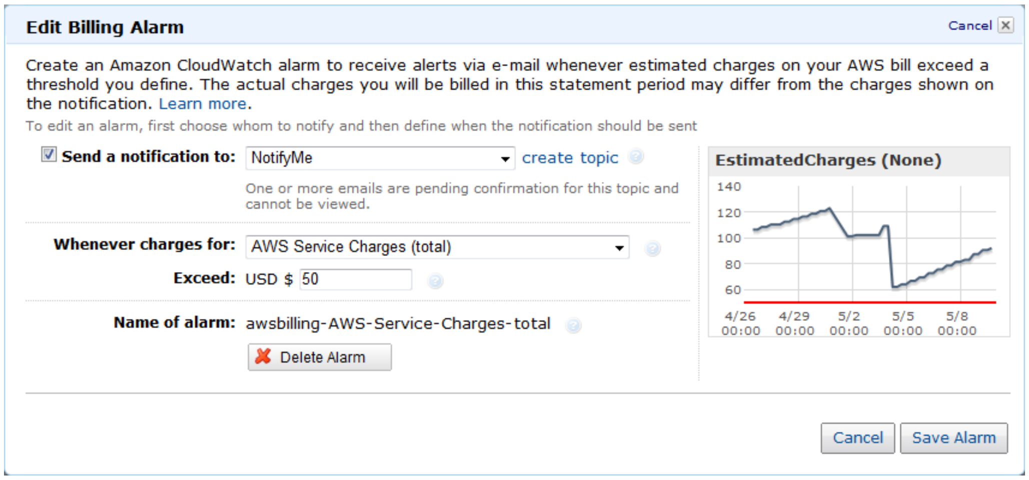
- At a command prompt, type `mon-describe-alarms --headers`, press Enter, and then in the list, locate the AWS/Billing alarm you want to check.

Deleting a Billing Alarm

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

To delete a billing alarm using the Amazon CloudWatch console

1. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the **Navigation** pane, click **Billing Alarms**.
3. In the **Alarm details for AWS billing estimated charges** dialog box, click the alarm you want to delete.
4. In the **Edit Billing Alarm** dialog box, click **Delete Alarm**.


Edit Billing Alarm

Create an Amazon CloudWatch alarm to receive alerts via e-mail whenever estimated charges on your AWS bill exceed a threshold you define. The actual charges you will be billed in this statement period may differ from the charges shown on the notification. Learn more.

To edit an alarm, first choose whom to notify and then define when the notification should be sent

Send a notification to: NotifyMe

One or more emails are pending confirmation for this topic and cannot be viewed.

Whenever charges for: AWS Service Charges (total)

Exceed: USD $ 50

Name of alarm: awsbilling-AWS-Service-Charges-total

To delete a billing alarm using the CLI

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

Publishing Custom Metrics

You can publish your own metrics to Amazon CloudWatch with the `mon-put-data` command (or its Query API equivalent `PutMetricData`). You can view statistical graphs of your published metrics with the AWS Management Console.

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

Note

The `mon-list-metrics` command can take up to two minutes to report metrics created by calls to `mon-put-data`. The Query API equivalent `ListMetrics`, however, can take up to fifteen minutes.

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

Publishing Single Data Points

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

```
mon-put-data --metric-name PageViewCount --namespace "MyService" --value 2 --
timestamp 2011-03-14T12:00:00.000Z
mon-put-data --metric-name PageViewCount --namespace "MyService" --value 4 --
timestamp 2011-03-14T12:00:01.000Z
mon-put-data --metric-name PageViewCount --namespace "MyService" --value 5 --
timestamp 2011-03-14T12:00:02.000Z
```

Note

If you want to run this example, specify time stamps within the past two weeks.

If you use the Query API instead of the command-line tools, you can publish up to 20 data points in a single call to `PutMetricData` by passing a list of data points. The list can contain data points that apply to different metrics, but the metrics must all belong to the same namespace.

Note

The `mon-put-data` command can currently publish only one data point per call.

Although you can publish data points with time stamps as granular as one-thousandth of a second, Amazon CloudWatch aggregates the data to a minimum granularity of one minute. For example, the `PageViewCount` metric from the previous examples contains three data points with time stamps just seconds apart. Amazon CloudWatch aggregates the three data points because they all have time stamps within a one-minute period.

Amazon CloudWatch uses one-minute boundaries when aggregating data points. For example, Amazon CloudWatch aggregates the data points from the previous example because all three data points fall within the one-minute period that begins at `2011-03-14T12:00:00.000Z` and ends at `2011-03-14T12:00:59.999Z`.

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

```
mon-get-stats PageViewCount -n "MyService" -s "Sum,Maximum,Minimum,Average,Sample
Count" --start-time 2011-03-14T12:00:00.000Z --end-time 2011-03-14T12:01:00.000Z
 --headers
```

Amazon CloudWatch returns the following:

```

Time                 SampleCount  Average            Sum   Minimum  Maximum
Unit
2011-03-14 12:00:00  3.0          3.6666666666666665 11.0  2.0      5.0
None
```

Publishing Statistic Sets

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

```
mon-put-data --metric-name PageViewCount --namespace "MyService" -s
"Sum=11,Minimum=2,Maximum=5,SampleCount=3" --timestamp 2011-03-14T12:00:00.000Z
```

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. You might want to publish zero instead of no value if you use periodic calls to `PutMetricData` to monitor the health of your application. For example, you can set an Amazon 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.

Amazon CloudWatch Command Line Interface Reference

Topics

This section describes the Amazon CloudWatch command line tool.

The command line tool is available as a ZIP file on the Amazon CloudWatch Developer Tools website. These tools are written in Java and include shell scripts for both Windows 2000/XP and Linux/UNIX/Mac OSX. The ZIP file is self-contained; no installation is required. You just download it and unzip it.

For more information about downloading and installing the Command Line Tool, see Command Line Tools (p. 13).

mon-cmd Command

This command lists the other Amazon CloudWatch commands.

```
Command Name                       Description

------------                       -----------

help
mon-delete-alarms                  Delete alarms

mon-describe-alarm-history         Describe alarm history

mon-describe-alarms                Describe alarms fully.

mon-describe-alarms-for-metric     Describe alarms associated with a metric

mon-disable-alarm-actions          Disable all actions for a given alarm

mon-enable-alarm-actions           Enable all actions for a given alarm

mon-get-stats                      Get metric statistics

mon-list-metrics                   List user's metrics
mon-put-data                       Put metric data
mon-put-metric-alarm              Create a new alarm or update an existing one

mon-set-alarm-state                Manually set the state of an alarm

version                            Prints the version of the CLI tool and API

For help on a specific command, type '<commandname> --help'
```

mon-delete-alarms Command

Delete the alarms with the specified name(s).

```
SYNOPSIS
mon-delete-alarms
[AlarmNames [AlarmNames ...] ]  [General Options]

DESCRIPTION
Delete alarms

ARGUMENTS
AlarmNames
Names of the alarms to delete. You can also set this value using "--alarm-name".

SPECIFIC OPTIONS
-f, --force
Forces the delete to go through without prompting. By default, the delete
command will prompt.


GENERAL OPTIONS
```

```
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Delete an alarm
```

```
$PROMPT> mon-delete-alarms --alarm-name my-alarm
```

mon-describe-alarm-history Command

Provides summary or detailed history information on the specified alarm.

```
SYNOPSIS
mon-describe-alarm-history
[AlarmName] [--end-date  value ] [--history-item-type  value ]
[--start-date  value ]  [General Options]

DESCRIPTION
Describe alarm history

ARGUMENTS
AlarmName
Names of the alarm.  By default history for all alarms will be returned.
You can also set this value using "--alarm-name".

SPECIFIC OPTIONS
--end-date VALUE
End of date range for history.   By default, it is current time.

--history-item-type VALUE
Type of history items you want to retrieve: one of ConfigurationUpdate,
StateUpdate or Action.  By default, all types will be returned.

--start-date VALUE
Start of date range for history.  By default it extends to all available
history.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.
```

```
-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Describe all history items for the alarm my-alm

$PROMPT> mon-describe-alarm-history --alarm-name my-alm --headers


OUTPUT
This command returns a table that contains the following:
* ALARM - Alarm name.
* TIMESTAMP - Timestamp.
* TYPE - Type of event, one of ConfigurationUpdate, StateUpdate and Action.
* SUMMARY - Human readable summary of history event.
* DATA - Detailed data on event in machine readable JSON format. This column
appears only in the --show-long view.

OUTPUT EXAMPLES

This is an example output of this command.

ALARM   TIMESTAMP                 TYPE                 SUMMARY
my-alm  2010-05-07T18:46:16.121Z  Action               Published a notification
 to arn:aws:sns:...
my-alm  2010-05-07T18:46:16.118Z  StateUpdate          Alarm updated from INSUF
FICIENT_DATA to OK
my-alm  2010-05-07T18:46:07.362Z  ConfigurationUpdate  Alarm "my-alm" created
```

mon-describe-alarms Command

Provides information on the specified alarm(s).

```
SYNOPSIS
mon-describe-alarms
[AlarmNames [AlarmNames ...] ] [--action-prefix  value ]
[--alarm-name-prefix  value ] [--state-value  value ]  [General Options]

DESCRIPTION
Describe alarms fully.

ARGUMENTS
AlarmNames
Names of the alarms. You can also set this value using "--alarm-name".

SPECIFIC OPTIONS
--action-prefix VALUE
Prefix of action names.

--alarm-name-prefix VALUE
Prefix of alarm names.

--state-value VALUE
State of Alarm.  One of OK, ALARM or INSUFFICIENT_DATA.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
```

```
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Describe all of your alarms whose name starts with my-alm

$PROMPT> mon-describe-alarms --alarm-name-prefix my-alm --headers


OUTPUT
This command returns a table that contains the following:
* ALARM - Alarm name.
* DESCRIPTION - Alarm description. This column appears only in the
--show-long view.
* STATE - Alarm state.
* STATE_REASON - Human readable reason for state. This column appears only
in the --show-long view.
* STATE_REASON_DATA - Machine readable reason for state (JSON format). This
column appears only in the --show-long view.
* ENABLED - Actions enabled or not. This column appears only in the
--show-long view.
* OK_ACTIONS - Action to execute on OK status. This column appears only in
the --show-long view.
* ALARM_ACTIONS - Action to execute on ALARM status.
* INSUFFICIENT_DATA_ACTIONS - Action to execute on INSUFFICIENT_DATA status.
This column appears
only in the --show-long view.
* NAMESPACE - Namespace for metric.
* METRIC_NAME - Metric name.
* DIMENSIONS - Dimensions. This column appears only in the --show-long view.
* PERIOD - Period.
* STATISTIC - Statistic.
* UNIT - Unit. This column appears only in the --show-long view.
* EVAL_PERIODS - Number of periods for which metric will be evaluated.
* COMPARISON - Comparison operator.
* THRESHOLD - Threshold.

OUTPUT EXAMPLES

This is an example output of this command.

ALARM    STATE ALARM_ACTIONS  NAMESPACE  METRIC_NAME    PERIOD  STATISTIC  EV
AL_PERIODS  COMPARISON            THRESHOLD
my-alm1  OK    arn:aws:sns:.. AWS/EC2    CPUUtilization 60      Average    3
           GreaterThanThreshold  100.0
my-alm2  OK    arn:aws:sns:.. AWS/EC2    CPUUtilization 60      Average    5
           GreaterThanThreshold  80.0
```

mon-describe-alarms-for-metric Command

Provide information on alarms associated with the specified metric.

```
SYNOPSIS
mon-describe-alarms-for-metric
--metric-name  value  --namespace  value [--alarm-description  value ]
[--dimensions  "key1=value1,key2=value2..." ] [--period  value ]
[--statistic  value ] [--unit  value ]  [General Options]

DESCRIPTION
Describe all alarms associated with a single metric

SPECIFIC OPTIONS
--alarm-description VALUE
No description available for this parameter.

--dimensions "key1=value1,key2=value2..."
Dimensions of the metric on which to alarm.

--metric-name VALUE
The name of the metric on which to alarm. Required.

--namespace VALUE
Namespace of the metric on which to alarm. Required.

--period VALUE
Period of metric on which to alarm.

--statistic VALUE
The statistic of the metric on which to alarm.    Possible values are
SampleCount, Average, Sum, Minimum or Maximum.

--unit VALUE
The unit of the metric on which to alarm.  Optional.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
```

```
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Describe an alarm for a given metric

$PROMPT>  mon-describe-alarms-for-metric --metric-name CPUUtilization --namespace
 AWS/EC2  --dimensions InstanceId=i-abcdef


OUTPUT
```

```
This command returns a table that contains the following:
* ALARM - Alarm name.
* DESCRIPTION - Alarm description. This column appears only in the
--show-long view.
* STATE - Alarm state.
* STATE_REASON - Human readable reason for state. This column appears only
in the --show-long view.
* STATE_REASON_DATA - Machine readable reason for state (JSON format). This
column appears only in the --show-long view.
* ENABLED - Actions enabled or not. This column appears only in the
--show-long view.
* OK_ACTIONS - Action to execute on OK status. This column appears only in
the --show-long view.
* ALARM_ACTIONS - Action to execute on ALARM status.
* INSUFFICIENT_DATA_ACTIONS - Action to execute on INSUFFICIENT_DATA status.
This column appears
only in the --show-long view.
* NAMESPACE - Namespace for metric.
* METRIC_NAME - Metric name.
* DIMENSIONS - Dimensions. This column appears only in the --show-long view.
* PERIOD - Period.
* STATISTIC - Statistic.
* UNIT - Unit. This column appears only in the --show-long view.
* EVAL_PERIODS - Number of periods for which metric will be evaluated.
* COMPARISON - Comparison operator.
* THRESHOLD - Threshold.

OUTPUT EXAMPLES

This is an example output of this command.

ALARM    STATE ALARM_ACTIONS  NAMESPACE  METRIC_NAME    PERIOD  STATISTIC  EV
AL_PERIODS  COMPARISON            THRESHOLD
my-alm1  OK    arn:aws:sns:.. AWS/EC2    CPUUtilization 60      Average    3
           GreaterThanThreshold  100.0
my-alm2  OK    arn:aws:sns:.. AWS/EC2    CPUUtilization 60      Average    5
           GreaterThanThreshold  80.0
```

mon-disable-alarm-actions Command

Disable all actions for the specified alarm(s).

```
SYNOPSIS
mon-disable-alarm-actions
[AlarmNames [AlarmNames ...] ]  [General Options]

DESCRIPTION
Disable all actions for a given alarm

ARGUMENTS
AlarmNames
List of alarm names. You can also set this value using "--alarm-name".

GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
```

```
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Disable all actions for an alarm

$PROMPT> mon-disable-alarm-actions --alarm-name my-alarm
```

mon-enable-alarm-actions Command

Enable all actions for a specified alarm(s).

```
SYNOPSIS
mon-enable-alarm-actions
[AlarmNames [AlarmNames ...] ]  [General Options]

DESCRIPTION
Enable all actions for a given alarm

ARGUMENTS
AlarmNames
List of alarm names. You can also set this value using "--alarm-name".

GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.
```

```
--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Enable all actions for an alarm

$PROMPT> mon-enable-alarm-actions --alarm-name my-alarm
```

mon-get-stats Command

Provide statistics for a given metric.

```
SYNOPSIS
mon-get-stats
MetricName  --namespace  value  --statistics  value[,value...]
[--dimensions  "key1=value1,key2=value2..." ] [--end-time  value ]
[--period  value ] [--start-time  value ] [--unit  value ]
[General Options]

DESCRIPTION
This call will get time-series data for one or more statistics of a given
MetricName.

ARGUMENTS
MetricName
The metric name that corresponds to one contained in the gathered Metric.
You can also set this value using "--metric-name". Required.

SPECIFIC OPTIONS
--dimensions "key1=value1,key2=value2..."
Dimensions (one or more) along which the metric data was originally
stored. If no dimensions are specified, then the statistics belonging to
the specified non-dimensional metric will be returned.

--end-time VALUE
The timestamp of the last datapoint to return, inclusive. For example,
2009-11-25T19:00:00+00:00. Timestamp will be rounded down to the nearest
minute.The dateTime type uses ISO 8601. The default for this is now.

-n, --namespace VALUE
The namespace of the desired metric.  This must match the namespace that
was specified when the desired metric was initially reported. Required.

--period VALUE
The granularity (in seconds) of the returned datapoints. Period must be
```

```
at least 60 seconds and must be a multiple of 60. The default is 60
seconds.

-s, --statistics VALUE1,VALUE2,VALUE3...
The statistics to be returned for the desired metric.  Valid values are:
Average, Sum, Maximum, or Minimum. Required.

--start-time VALUE
The timestamp of the first datapoint to return, inclusive. For example,
2009-11-25T19:00:00+00:00. Timestamp will be rounded down to the nearest
minute. The dateTime type uses ISO 8601. The default for this is 1 hour
in the past.

--unit VALUE
The unit that the metric was reported in.  Valid unit values are Seconds,
Bytes, Bits, Percent, Count, Bytes/Second, Bits/Second, Count/Second,
None.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.
```

```
--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

This returns the average, min, and max CPU utilization for EC2 instance
i-c07704a9, at 1 hour resolution.

$PROMPT>mon-get-stats CPUUtilization --start-time 2009-02-14T23:00:00.000Z --
end-time 2009-03-14T23:00:00.000Z --period 3600 --statistics "Average,Minim
um,Maximum" --namespace "AWS/EC2" --dimensions "InstanceId=i-c07704a9"


This returns CPU utilization across your EC2 fleet.

$PROMPT2>mon-get-stats CPUUtilization --start-time 2009-02-14T23:00:00.000Z --
end-time 2009-03-14T23:00:00.000Z --period 3600 --statistics "Average,Minim
um,Maximum" --namespace "AWS/EC2"


This returns the average, min, and max Request count made to the test stack
of "MyService" for a particular user, at 1 hour resolution.

$PROMPT>mon-get-stats RequestCount --start-time 2009-11-24T23:00:00.000Z --end-
time 2009-11-25T23:00:00.000Z --period 3600 --statistics "Average,Minimum,Max
imum" --namespace "MyService" --dimensions "User=SomeUser,Stack=Test"


This shows RequestCount statistics across all of "MyService".

$PROMPT2>mon-get-stats RequestCount --start-time 2009-11-24T23:00:00.000Z --
end-time 2009-11-25T23:00:00.000Z --period 3600 --statistics "Average,Minim
um,Maximum,SampleCount" --namespace "MyService"


OUTPUT
This command returns a table that contains the following:
* Time - Time the metrics were taken.
* SampleCount - No description available for this column.
* Average - Average value.
* Sum - Sum of values.
* Minimum - Minimum observed value.
* Maximum - Maximum observed value.
* Unit - Unit of the metric.
```

```
OUTPUT EXAMPLES

This is an example of an output of the Samples and Average metrics at one
minute resolution.

Time                 Samples  Average  Unit
2009-05-19 00:03:00  2.0      0.19     Percent
2009-05-19 00:04:00  2.0      0        Percent
2009-05-19 00:05:00  2.0      0        Percent
2009-05-19 00:06:00  2.0      0        Percent
2009-05-19 00:07:00  2.0      0        Percent
2009-05-19 00:08:00  2.0      0        Percent
2009-05-19 00:09:00  2.0      0        Percent
2009-05-19 00:10:00  2.0      0        Percent
2009-05-19 00:11:00  2.0      0        Percent
2009-05-19 00:12:00  2.0      0.195    Percent
2009-05-19 00:13:00  2.0      0.215    Percent
...
```

mon-list-metrics Command

List the metrics associated with your AWS account. You can filter metrics by using any combination of MetricName, Namespace, or Dimensions. If a you do not specify a filter, all possible matches for the attribute are returned.

Note

Any metric not seen for two weeks will expire and be removed from from this list.

Note

The `mon-list-metrics` command can take up to two minutes to report data points added by calls to `mon-put-data`. The Query API equivalent `ListMetrics`, however, can take up to fifteen minutes.

```
SYNOPSIS
mon-list-metrics
[--dimensions  "key1=value1,key2=value2..." ] [--metric-name  value ]
[--namespace  value ]  [General Options]

DESCRIPTION
This call will get a list of all the metrics that the system is storing for
your AWS account. Note: Any metric not seen for two weeks will be aged out
from this list as that is the retention period for data. Metrics can be
filtered by any combination of MetricName, Namespace or Dimensions.  If any
of these filters is not specified, then all possible matches for the
respective attribute are returned.

SPECIFIC OPTIONS
-d, --dimensions "key1=value1,key2=value2..."
Dimensions (one or more) along which the metric names are to be filtered.
If no dimensions are specified then metric names with all possible
dimensions will be included in the results.

-m, --metric-name VALUE
The name of a particular metric that you are interested in.  If not
```

```
specified, all possible metric names will be returned.

-n, --namespace VALUE
The namespace along which metrics will be filtered.  If not specified,
metric names from  all possible namespaces will be returned.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
```

```
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

This returns a list of all your metrics.

$PROMPT>mon-list-metrics


This returns a list of all your metrics that have a particular metric name.

$PROMPT>mon-list-metrics --metric-name RequestCount


This returns a list of all your metrics that belong to a particular
namespace

$PROMPT>mon-list-metrics --namespace MyService


This returns a list of all your metrics having the specified dimension names
and values.

$PROMPT>mon-list-metrics --dimensions "User=SomeUser,Stack=Test"


OUTPUT
This command returns a table that contains the following:
* Metric Name - The name of the metric attached to this metric.
* Namespace - The namespace associated with this metric.
* Dimensions - A list of the dimension names and values associated with this
  metric.

OUTPUT EXAMPLES

This is an example of an output of a call to 'mon-list-metrics'.

Metric Name                 Namespace  Dimensions
CPUUtilization               AWS/EC2    {InstanceId=i-e7e48a8e}
CPUUtilization               AWS/EC2    {InstanceId=i-231d744a}
CPUUtilization               AWS/EC2    {InstanceId=i-22016e4b}
CPUUtilization               AWS/EC2    {InstanceId=i-b0345cd9}
CPUUtilization               AWS/EC2    {InstanceId=i-539dff3a}
CPUUtilization               AWS/EC2    {InstanceId=i-af3544c6}
CPUUtilization               AWS/EC2    {InstanceId=i-d4f29ebd}
CPUUtilization               AWS/EC2    {ImageId=ami-de4daab7}
...
```

mon-put-data Command

Add metric data points to a metric.

Note

The `mon-list-metrics` command can take up to two minutes to report data points added by calls to `mon-put-data`. The Query API equivalent `ListMetrics`, however, can take up to fifteen minutes.

```
SYNOPSIS
mon-put-data
--metric-name  value[,value...]  --namespace  value [--dimensions
"key1=value1,key2=value2..." ] [--statisticValues
"key1=value1,key2=value2..." ] [--timestamp  value[,value...] ] [--unit
value[,value...] ] [--value  value[,value...] ]  [General Options]

DESCRIPTION
This call will put time-series data, for either the raw value  or valid
statistic values of a given MetricName.  It supports the input of a single
datapoint at a time.

SPECIFIC OPTIONS
-d, --dimensions "key1=value1,key2=value2..."
Dimensions (one or more) along which metric data can be uniquely
identified.

-m, --metric-name VALUE1,VALUE2,VALUE3...
The name of the Metric to be gathered. Required.

-n, --namespace VALUE
The namespace the given Metric is valid in. Required.

-s, --statisticValues "key1=value1,key2=value2..."
The statistics to be provided for the given metric. Valid key values are:
SampleCount, Sum, Maximum, and Minimum.  All these values need to be
specified for a valid call.

-t, --timestamp VALUE1,VALUE2,VALUE3...
The timestamp of the datapoint. For example, 2009-11-25T19:00:00+00:00.
Timestamp will be rounded down to the nearest minute. The dateTime type
uses ISO 8601. The default for this is the time that this command was
executed.

-u, --unit VALUE1,VALUE2,VALUE3...
The unit that the metric is being reported in.  Valid unit values are
Seconds, Bytes, Bits, Percent, Count, Bytes/Second, Bits/Second,
Count/Second, None.

-v, --value VALUE1,VALUE2,VALUE3...
The value of the metric datum being put in.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.
```

```
--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

This puts statistic data for "RequestCount" in the "MyService" namespace.
The metric contains no dimensions and so represents the overall RequestCount
across the entire service.  The measurement is a pre-aggregated
statisticValue representing five earlier measurements whose maximum was 70,
minimum was 30 and sum was 250.

$PROMPT>mon-put-data --metric-name RequestCount --namespace "MyService" --
timestamp 2009-11-25T00:00:00.000Z --statisticValues "Sum=250,Minimum=30,Maxim
um=70,SampleCount=5"
```

```

This puts user-specific "RequestCount" test data in the "MyService"
namespace.  The user and stack name are stored as dimensions in order to
distinguish this metric from the service-wide metric in the example above.

$PROMPT>mon-put-data --metric-name RequestCount --namespace "MyService" --dimen
sions "User=SomeUser,Stack=Test" --timestamp 2009-11-25T00:00:00.000Z --value
50
```

mon-put-metric-alarm Command

Create a new alarm, or update an existing one.

```
SYNOPSIS
mon-put-metric-alarm
AlarmName  --comparison-operator  value  --evaluation-periods  value
--metric-name  value  --namespace  value  --period  value  --statistic
value  --threshold  value [--actions-enabled  value ] [--alarm-actions
value[,value...] ] [--alarm-description  value ] [--dimensions
"key1=value1,key2=value2..." ] [--ok-actions  value[,value...] ] [--unit
value ] [--insufficient-data-actions  value[,value...] ]  [General Options]

DESCRIPTION
Create a new alarm or update an existing one

ARGUMENTS
AlarmName
Name of the alarm. You can also set this value using "--alarm-name".
Required.

SPECIFIC OPTIONS
--actions-enabled VALUE
Should actions be executed when this alarm changes state   (true or
false).

--alarm-actions VALUE1,VALUE2,VALUE3...
SNS topics to which notification should be sent if the alarm goes to
state ALARM.

--alarm-description VALUE
Description of alarm.

--comparison-operator VALUE
The operator with which the comparison with threshold will be made: one
of  GreaterThanOrEqualToThreshold, GreaterThanThreshold,
LessThanThreshold and LessThanOrEqualToThreshold. Required.

--dimensions "key1=value1,key2=value2..."
Dimensions of the metric on which to alarm.

--evaluation-periods VALUE
Number of consecutive periods for which the value of the metric needs to
be compared to threshold. Required.

--metric-name VALUE
The name of the metric on which to alarm. Required.
```

```
--namespace VALUE
Namespace of the metric on which to alarm. Required.

--ok-actions VALUE1,VALUE2,VALUE3...
SNS topics to which notification should be sent if the alarm goes to
state OK.

--period VALUE
Period of metric on which to alarm. Required.

--statistic VALUE
The statistic of the metric on which to alarm.    Possible values are
SampleCount, Average, Sum, Minimum, Maximum. Required.

--threshold VALUE
The threshold with which the metric value will be compared. Required.

--unit VALUE
The unit of the metric on which to alarm.

--insufficient-data-actions VALUE1,VALUE2,VALUE3...
SNS topics to which notification should be sent if the alarm goes to
state INSUFFICIENT_DATA.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.
```

```
-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Create an alarm my-alarm which publishes a message to a topic when CPU
utilization of an EC2 instances exceeds 90% for 3 consecutive 1 min periods.

$PROMPT>  mon-put-metric-alarm --alarm-name my-alarm --alarm-description "some
 desc" --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average  -
-period 60 --threshold 90 --comparison-operator GreaterThanThreshold  --dimen
sions InstanceId=i-abcdef --evaluation-periods 3  --unit Percent --alarm-actions
 arn:aws:sns:us-east-1:1234567890:my-topic
```

mon-set-alarm-state Command

Temporarily change the alarm state. On the next period, the alarm is set to its true state.

```
SYNOPSIS
mon-set-alarm-state
AlarmName  --state-reason  value  --state-value  value
[--state-reason-data  value ]  [General Options]

DESCRIPTION
Manually set the state of an alarm

ARGUMENTS
AlarmName
Name of the alarm. You can also set this value using "--alarm-name".
Required.

SPECIFIC OPTIONS
--state-reason VALUE
The reason why this alarm was set to this state (human readable).
Required.
```

```
--state-reason-data VALUE
The reason why this alarm was set to this state (machine readable JSON).

--state-value VALUE
State to be set: one of ALARM, OK or INSUFFICIENT_DATA. Required.


GENERAL OPTIONS
--aws-credential-file VALUE
Location of the file with your AWS credentials. This value can be set by
using the environment variable 'AWS_CREDENTIAL_FILE'.

-C, --ec2-cert-file-path VALUE
Location of your EC2 certificate file. This value can be set by using the
environment variable 'EC2_CERT'.

--connection-timeout VALUE
Specify a connection timeout VALUE (in seconds). The default value is
'30'.

--delimiter VALUE
What delimiter to use when displaying delimited (long) results.

--headers
If you are displaying tabular or delimited results, it includes the
column headers. If you are showing xml results, it returns the HTTP
headers from the service request, if applicable. This is off by default.

-I, --access-key-id VALUE
Specify VALUE as the AWS Access Id to use.

-K, --ec2-private-key-file-path VALUE
Location of your EC2 private key file. This value can be set by using the
environment variable 'EC2_PRIVATE_KEY'.

--region VALUE
Specify region VALUE as the web service region to use. This value can be
set by using the environment variable 'EC2_REGION'.

-S, --secret-key VALUE
Specify VALUE as the AWS Secret Key to use.

--show-empty-fields
Show empty fields using "(nil)" as a placeholder to indicate that this data was
not requested. Empty fields are not shown by default.

--show-request
Displays the URL the tools used to call the AWS Service. The default
value is 'false'.

--show-table, --show-long, --show-xml, --quiet
Specify how the results are displayed: tabular, delimited (long), xml, or
no output (quiet). Tabular shows a subset of the data in fixed
column-width form, while long shows all of the returned values delimited
by a character. The xml is the raw return from the service, while quiet
suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url VALUE
```

```
This option will override the URL for the service call with VALUE. This
value can be set by using the environment variable 'AWS_CLOUDWATCH_URL'.


INPUT EXAMPLES

Set the state of alarm to OK

$PROMPT> mon-set-alarm-state --alarm-name my-alarm --state OK
```

mon-version Command

Print the Amazon CloudWatch version number.

```
SYNOPSIS
   mon-version

DESCRIPTION
   Prints the version of the CLI tool and the API.
```

Amazon CloudWatch Monitoring Scripts

The Amazon CloudWatch Monitoring Scripts for Linux and Windows demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts for Linux are sample Perl scripts that comprise a fully functional example that reports memory, swap, and disk space utilization metrics for an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance. The scripts for Windows are sample PowerShell scripts that comprise a fully functional example that reports memory, page file, and disk space utilization metrics for an Amazon EC2 Windows instance. You can download the Amazon CloudWatch Monitoring Scripts for Linux and for Windows from the Amazon Web Services (AWS) sample code library and install them on your Linux- or Windows-based instances.

Important

These scripts are examples only. They are provided "as is" and are not supported.

Note

Standard Amazon CloudWatch free tier quantities and usage charges for custom metrics apply to your use of these scripts. For more information, see the Amazon CloudWatch product page.

Topics

Amazon CloudWatch Monitoring Scripts for Linux

The Amazon CloudWatch Monitoring Scripts for Linux are sample Perl scripts that demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts comprise a fully functional example that reports memory, swap, and disk space utilization metrics for an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance.

Topics

You can download the Amazon CloudWatch Monitoring Scripts for Linux from the AWS sample code library. The CloudWatchMonitoringScripts.zip package contains these files:

- **CloudWatchClient.pm**—Shared Perl module that simplifies calling Amazon CloudWatch from other scripts.
- **mon-put-instance-data.pl**—Collects system metrics on an Amazon EC2 instance (memory, swap, disk space utilization) and sends them to Amazon CloudWatch.
- **mon-get-instance-stats.pl**—Queries Amazon CloudWatch and displays the most recent utilization statistics for the EC2 instance on which this script is executed.
- **awscreds.conf**—File template for AWS credentials that stores your access key ID and secret access key.
- **LICENSE.txt**—Text file containing the Apache 2.0 license.
- **NOTICE.txt**—copyright notice.

These monitoring scripts are intended for use with Amazon EC2 instances running Linux operating systems. The scripts have been tested on the following Amazon Machine Images (AMIs) for both 32-bit and 64-bit versions:

- Amazon Linux 2011.09.02
- Red Hat Enterprise Linux 6.2
- Ubuntu Server 11.10
- SUSE Linux Enterprise Server 11

Prerequisites

No additional steps are required on most versions of Linux. However, if you are running an Ubuntu Server, use the following procedure to configure your server. Log on to your Ubuntu instance and install the following packages:

```
sudo apt-get install unzip libwww-perl libcrypt-ssleay-perl
```

For information on how to connect to Amazon EC2 Linux instances, see Connecting to Instances in the *Amazon EC2 User Guide*.

Getting Started

The following steps show you how to download, uncompress, and configure the CloudWatch Monitoring Scripts on an EC2 Linux instance.

To download, install, and configure the script

1. Open a command prompt and enter the following:

   ```
   cd
   mkdir aws-scripts-mon
   cd aws-scripts-mon
   ```

```
wget http://ec2-downloads.s3.amazonaws.com/cloudwatch-samples/CloudWatchMon
itoringScripts.zip
unzip CloudWatchMonitoringScripts.zip
rm CloudWatchMonitoringScripts.zip
```

2. Update the `awscreds.conf` file that you downloaded earlier. The content of this file should use the following format:

   `AWSAccessKeyId=`*`YourAccessKeyID`*

   `AWSSecretKey=`*`YourSecretAccessKey`*

   Note

   This step is optional if you have already created a file for credentials. You can use an existing file by specifying its location on the command line when you call the scripts. Alternatively, you can set the environment variable `AWS_CREDENTIAL_FILE` to point to the file with your AWS credentials.

   For instructions on how to access your credentials, use the following procedure.

To view your AWS access credentials

1. Go to the Amazon Web Services website at http://aws.amazon.com.
2. Click **My Account/Console**, and then click **Security Credentials**.
3. Under **Your Account**, click **Security Credentials**.
4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.
5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

As a best practice, do not use the root credentials. Instead you should create an Identity and Access Management (IAM) user with a policy that restricts the user to only Amazon CloudWatch operations. For more information, see Controlling User Access to Your AWS Account (p. 137)

Using the Scripts

mon-put-instance-data.pl

This script collects memory, swap, and disk space utilization data on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
`--mem-util`	Collects and sends the MemoryUtilization metrics in percentages. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
`--mem-used`	Collects and sends the MemoryUsed metrics, reported in megabytes. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.

Name	Description
--mem-avail	Collects and sends the MemoryAvailable metrics, reported in megabytes. This option reports memory available for use by applications and the operating system.
--swap-util	Collects and sends SwapUtilization metrics, reported in percentages.
--swap-used	Collects and sends SwapUsed metrics, reported in megabytes.
--disk-path=PATH	Selects the disk on which to report. PATH can specify a mount point or any file located on a mount point for the filesystem that needs to be reported. For selecting multiple disks, specify a `--disk-path=PATH` for each one of them. To select a disk for the filesystems mounted on `/` and `/home`, use the following parameters: `--disk-path=/ --disk-path=/home`
--disk-space-util	Collects and sends the DiskSpaceUtilization metric for the selected disks. The metric is reported in percentages.
--disk-space-used	Collects and sends the DiskSpaceUsed metric for the selected disks. The metric is reported by default in gigabytes. Due to reserved disk space in Linux operating systems, disk space used and disk space available may not accurately add up to the amount of total disk space.
--disk-space-avail	Collects and sends the DiskSpaceAvailable metric for the selected disks. The metric is reported in gigabytes. Due to reserved disk space in the Linux operating systems, disk space used and disk space available may not accurately add up to the amount of total disk space.
--memory-units=UNITS	Specifies units in which to report memory usage. If not specified, memory is reported in megabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
--disk-space-units=UNITS	Specifies units in which to report disk space usage. If not specified, disk space is reported in gigabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
--aws-credential-file=PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the `--aws-access-key-id` and `--aws-secret-key` parameters.
--aws-access-key-id=VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the `--aws-secret-key` option. Do not use this option with the `--aws-credential-file` option.
--aws-secret-key=VALUE	Specifies the AWS secret access key to use to sign the request to CloudWatch. Must be used together with the `--aws-access-key-id` option. Do not use this option with `--aws-credential-file` option.

Name	Description
--verify	Performs a test run of the script that collects the metrics, prepares a complete HTTP request, but does not actually call CloudWatch to report the data. This option also checks that credentials are provided. When run in verbose mode, this option outputs the metrics that will be sent to Amazon CloudWatch.
--from-cron	Use this option when calling the script from cron. When this option is used, all diagnostic output is suppressed, but error messages are sent to the local system log of the user account.
--verbose	Displays detailed information about what the script is doing.
--help	Displays usage information.
--version	Displays the version number of the script.

Examples

The following examples assume that you have already updated the `awscreds.conf` file with valid AWS credentials. If you are not using the `awscreds.conf` file, provide credentials using the `--aws-access-key-id` and `--aws-secret-key` arguments.

To perform a simple test run without posting data to Amazon CloudWatch

- Run the following command:

```
./mon-put-instance-data.pl --mem-util --verify --verbose
```

To collect all available memory metrics and send them to Amazon CloudWatch

- Run the following command:

```
./mon-put-instance-data.pl --mem-util --mem-used --mem-avail
```

To set a cron schedule for metrics reported to Amazon CloudWatch

1. Start editing the crontab using the following command:

```
crontab -e
```

2. Add the following command to report memory and disk space utilization to CloudWatch every five minutes:

```
*/5 * * * * ~/aws-scripts-mon/mon-put-instance-data.pl --mem-util --disk-
space-util --disk-path=/ --from-cron
```

If the script encounters an error, the script will write the error message in the system log.

mon-get-instance-stats.pl

This script queries Amazon CloudWatch for statistics on memory, swap, and disk space metrics within the time interval provided using the number of most recent hours. This data is provided for the Amazon EC2 instance on which this script is executed.

Options

Name	Description
--recent-hours=N	Specifies the number of recent hours to report on, as represented by `N` where N is an integer.
--aws-credential-file=PATH	Provides the location of the file containing AWS credentials.
--aws-access-key-id=VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the `--aws-secret-key` option. Do not use this option with the `--aws-credential-file` option.
--verify	Performs a test run of the script that collects the metrics, prepares a complete HTTP request, but does not actually call CloudWatch to report the data. This option also checks that credentials are provided. When run in verbose mode, this option outputs the metrics that will be sent to Amazon CloudWatch.
--verbose	Displays detailed information about what the script is doing.
--help	Displays usage information.
--version	Displays the version number of the script.

Examples

To get utilization statistics for the last 12 hours

- Run the following command:

```
mon-get-instance-stats.pl --recent-hours=12
```

The returned response will be similar to the following example output:

```
Instance metric statistics for the last 12 hours.

CPU Utilization
    Average: 1.06%, Minimum: 0.00%, Maximum: 15.22%

Memory Utilization
    Average: 6.84%, Minimum: 6.82%, Maximum: 6.89%

Swap Utilization
    Average: N/A, Minimum: N/A, Maximum: N/A

Disk Space Utilization on /dev/xvda1 mounted as /
    Average: 9.69%, Minimum: 9.69%, Maximum: 9.69%
```

Viewing Your Custom Metrics in the AWS Management Console

If you successfully call the mon-put-instance-data.pl script, you can use the AWS Management Console to view your posted custom metrics in the Amazon CloudWatch console.

To view custom metrics

1. Execute mon-put-instance-data.pl, as described earlier.
2. Sign in to the AWS Management Console and open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
3. Click **View Metrics**.
4. In the **Viewing** drop-down list, your custom metrics posted by the script display with the prefix System/Linux.

Amazon CloudWatch Monitoring Scripts for Windows

The Amazon CloudWatch Monitoring Scripts for Windows are sample PowerShell scripts that demonstrate how to produce and consume Amazon CloudWatch custom metrics. The scripts comprise a fully functional example that reports memory, page file, and disk space utilization metrics for an Amazon Elastic Compute Cloud (Amazon EC2) Windows instance.

Topics

You can download Amazon CloudWatch Monitoring Scripts for Microsoft Windows Server from the Amazon Web Services (AWS) sample code library. The AmazonCloudWatchMonitoringWindows.zip package contains these files:

- **mon-put-metrics-mem.ps1** —Collects system metrics on an Amazon EC2 Windows instance (memory, page file utilization) and sends them to Amazon CloudWatch.
- **mon-put-metrics-disk.ps1** —Collects system metrics on an Amazon EC2 instance (disk space utilization) and sends them to Amazon CloudWatch.
- **mon-put-metrics-perfmon.ps1** —Collects PerfMon counters on an Amazon EC2 instance and sends them to Amazon CloudWatch.
- **mon-get-instance-stats.ps1**—Queries Amazon CloudWatch and displays the most recent utilization statistics for the EC2 instance on which this script is executed.
- **awscreds.conf**—File template for AWS credentials that stores your access key ID and secret access key.
- **LICENSE.txt**—Text file containing the Apache 2.0 license.
- **NOTICE.txt**—Copyright notice.

These monitoring scripts are intended for use with Amazon EC2 instances running Microsoft Windows Server. The scripts have been tested on the following Amazon Machine Images (AMIs) for both 32-bit and 64-bit versions:

- Windows Server 2003 R2
- Windows Server 2008

- Windows Server 2008 R2

Getting Started

The following steps demonstrate how to download, un-compress, and configure the Amazon CloudWatch Monitoring Scripts on an Amazon EC2 Windows instance.

To download, install, and configure the script

1. Connect to your Amazon EC2 Windows instance. For information about how to connect to Amazon EC2 Windows instances, see Connecting to Windows Instances in the *Amazon EC2 User Guide*.
2. Download and install the AWS SDK for .NET onto the EC2 instance that you want to monitor.
3. Download the .zip file containing the Amazon CloudWatch Monitoring Scripts for Microsoft Windows Server onto the EC2 instance and unzip it in a location of your preference.
4. Update the awscreds.conf file that you downloaded earlier. The content of this file should use the following format:

   ```
   AWSAccessKeyId=YourAccessKeyID
   ```

   ```
   AWSSecretKey=YourSecretAccessKey
   ```

   Note

   This step is optional if you have already created a file for credentials. You can use an existing file by specifying its location on the command line when you call the scripts. Alternatively, you can set the environment variable `AWS_CREDENTIAL_FILE` to point to the file with your AWS credentials.

   For instructions on how to access your credentials, use the following procedure.

   To view your AWS access credentials

   1. Go to the Amazon Web Services website at http://aws.amazon.com.
   2. Click **My Account/Console**, and then click **Security Credentials**.
   3. Under **Your Account**, click **Security Credentials**.
   4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.
   5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

   As a best practice, do not use the root credentials. Instead you should create an Identity and Access Management (IAM) user with a policy that restricts the user to only Amazon CloudWatch operations. For more information, see Controlling User Access to Your AWS Account (p. 137)

Using the Scripts

mon-put-metrics-mem.ps1

This script collects memory and pagefile utilization data on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
-mem-util	Collects and sends the MemoryUtilization metrics in percentages. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
-mem-used	Collects and sends the MemoryUsed metrics, reported in megabytes. This option reports only memory allocated by applications and the operating system, and excludes memory in cache and buffers.
-mem-avail	Collects and sends the MemoryAvailable metrics, reported in megabytes. This option reports memory available for use by applications and the operating system.
-page-util	Collects and sends PageUtilization metrics, reported in percentages. Page utilization is reported for each page file in a windows instance.
-page-used	Collects and sends PageUsed metrics, reported in megabytes.
-page_avail	Reports available space in page file for all disks.
-memory_units UNITS	Specifies units in which to report memory usage. If not specified, memory is reported in megabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
-aws_credential_file=PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the `-aws_access_id` and `-aws_secret_key` parameters.
-aws_access_id=VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the `-aws_secret_key` option. Do not use this option with the `-aws_credential_file` option.
-aws_secret_key=VALUE	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the `-aws_access-key_id` option. Do not use this option with `-aws_credential_file` option.
-whatif	Performs a test run of the script that collects the metrics but does not actually call Amazon CloudWatch to report the data. This option also checks that credentials are provided.
-from_scheduler	Use this option when calling the script from task scheduler. When this option is used, all diagnostic output is suppressed, but error messages are sent to the log file.
-verbose	Displays detailed information about what the script is doing.
Get-help mon-put-metrics-mem.ps1	Displays usage information.
-version	Displays the version number of the script.
-logfile	Logfile is used to log error message. Use this along with –from_scheduler option. If no value is specified for logfile then a default file is created with the same as the script with .log extension.

Examples

The following examples assume that you have already updated the `awscreds.conf` file with valid AWS credentials. If you are not using the `awscreds.conf` file, provide credentials using the `-aws_access_id` and `-aws_secret_key` arguments.

To collect all available memory metrics using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-mem.ps1 -aws_access_id ThisIsMyAccessKey -aws_secret_key
 ThisIsMySecretKey -mem_util -mem_avail -page_avail -page_used -page_util
-memory_units Megabytes
```

To collect all available memory metrics using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-mem.ps1 -aws_credential_file C:\awscreds.conf -mem_util
-mem_used -mem_avail -page_avail -page_used -page_util -memory_units Megabytes
```

To collect all available memory metrics using credentials stored in environment variables and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-mem.ps1 -mem_util -mem_used -mem_avail -page_avail -
page_used -page_util -memory_units Megabytes
```

mon-put-metrics-disk.ps1

This script collects disk space utilization data on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
-disk_space_util	Collects and sends the DiskSpaceUtilization metric for the selected disks. The metric is reported in percentages.
-disk_space_used	Collects and sends DiskSpaceUsed metric for the selected disks. The metric is reported by default in gigabytes.
-disk_space_avail	Collects and sends the DiskSpaceAvailable metric for the selected disks. The metric is reported in gigabytes.

Name	Description
-disk_space_units UNITS	Specifies units in which to report memory usage. If not specified, memory is reported in gigabytes. UNITS may be one of the following: bytes, kilobytes, megabytes, gigabytes.
-disk_drive	Selects the drive letter on which to report. To report metric on c and d drive use the following option `-disk_drive` C:, D: Values should be comma separated.
-aws_credential_file PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the `–aws_access_id` and `-aws_secret_key` parameters.
-aws_access_id VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the `–aws_secret_key` option. Do not use this option with the `-aws_credential_file` option.
-aws_secret_key VALUE	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the `-aws_access_id` option. Do not use this option with `-aws_credential_file` option.
-whatif	Performs a test run of the script that collects the metrics but does not actually call Amazon CloudWatch to report the data. This option also checks that credentials are provided.
-from_scheduler	Use this option when calling the script from task scheduler. When this option is used, all diagnostic output is suppressed, but error messages are sent to the log file.
-verbose	Displays detailed information about what the script is doing.
Get-help mon-put-metrics-disk.ps1	Displays usage information.
-version	Displays the version number of the script.
-logfile	Logfile is used to log error message. Use this along with `–from_scheduler` option. If no value is specified for logfile then a default file is created with the same as the script with .log extension.

Examples

To collect all available disk metrics using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-disk.ps1  -aws_access_id ThisIsMyAccessKey -aws_secret_key
 ThisIsMySecretKey -disk_space_util -disk_space_avail -disk_space_units
Gigabytes
```

To collect all available disk metrics using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-disk.ps1
                  -aws_credential_file C:\awscreds.conf -disk_drive C:, d
-disk_space_util -disk_space_used -disk_space_avail -disk_space_units Giga
bytes
```

To collect all available disk metrics using credentials stored in an environment variable and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-disk.ps1  -disk_drive C:, d
                  -disk_space_util -disk_space_used -disk_space_avail -
disk_space_units Gigabytes
```

mon-put-metrics-perfmon.ps1

This script collects PerfMon counters on the current system. It then makes a remote call to Amazon CloudWatch to report the collected data as custom metrics.

Options

Name	Description
-processor_queue	Reports current processor queue counter.
-pages_input	Reports memory pages/input memory counter.
-aws_credential_file PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the `–aws_access_id` and `-aws_secret_key` parameters.
-aws_access_id VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the `–aws_secret_key` option. Do not use this option with the `-aws_credential_file` option.
-aws_secret_key VALUE	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the `-aws_access_id` option. Do not use this option with `-aws_credential_file` option.
-whatif	Performs a test run of the script that collects the metrics but does not actually call Amazon CloudWatch to report the data. This option also checks that credentials are provided.
-from_scheduler	Use this option when calling the script from task scheduler. When this option is used, all diagnostic output is suppressed, but error messages are sent to the log file.

Name	Description
-verbose	Displays detailed information about what the script is doing.
Get-help mon-put-metrics-disk.ps1	Displays usage information.
-version	Displays the version number of the script.
-logfile	Logfile is used to log error message. Use this along with `–from_scheduler` option. If no value is specified for logfile then a default file is created with the same as the script with .log extension.

Examples

To collect preset PerfMon counters in script using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-perfmon.ps1 -aws_access_id ThisIsMyAccessKey -aws_secret_key
 ThisIsMySecretKey -pages_input -processor_queue
```

To collect preset PerfMon counters in script using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-perfmon.ps1  -aws_credential_file C:\awscreds.conf -
pages_input -processor_queue
```

To collect preset PerfMon counters in script using credentials stored in an environment variable and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-put-metrics-perfmon.ps1 -pages_input -processor_queue
```

To add more counters to be pushed to Amazon CloudWatch

1. Open the script in a text editor such as Notepad, and then on line 72, locate the following commented section:

```
### Add More counters here.
#$Counters.Add('\Memory\Cache Bytes','Bytes')
#$Counters.Add('\\localhost\physicaldisk(0 c:)\% disk time','Percent')
```

Note

The first parameter (e.g., $Counters.Add) is the PerfMon counter. The second parameter (e.g., ('\Memory\Cache Bytes','Bytes')) is the unit of data that counter provides.

2. Edit the script and add your own PerfMon counters to the script as shown above. After you have added custom PerfMon counters to the script, you can run the script without any parameters other than credential information.

   Note

   You can only add PerfMon counters to the script on your computer. You can use the `Get-Counter` command to test PerfMon counters. For more information, see Get-Counter on the Microsoft TechNet website.

mon-get-instance-stats.ps1

This script queries Amazon CloudWatch for statistics on memory, page file, and disk space metrics within the time interval provided using the number of most recent hours. This data is provided for the Amazon EC2 instance on which this script is executed.

Options

Name	Description
-recent-hours N	Specifies the number of recent hours to report on, as represented by N where N is an integer.
-aws_credential_file PATH	Provides the location of the file containing AWS credentials. This parameter cannot be used with the `–aws_access_id` and `-aws_secret_key` parameters.
-aws_access_id VALUE	Specifies the AWS access key ID to use to identify the caller. Must be used together with the `–aws_secret_key` option. Do not use this option with the `-aws_credential_file` option.
-aws_secret_key VALUE	Specifies the AWS secret access key to use to sign the request to Amazon CloudWatch. Must be used together with the `-aws_access_id` option. Do not use this option with `-aws_credential_file` option.
-verbose	Displays detailed information about what the script is doing.
Get-help mon-get-instance-stats.ps1	Displays usage information.
-version	Displays the version number of the script.

Examples

To get utilization statistics for the last 12 hours using an inline access ID and secret key and send the data to Amazon CloudWatch

- Run the following command:

```
.\ mon-get-instance-stats.ps1 -aws_access_id
              ThisIsMyAccessKey -aws_secret_key ThisIsMySecretKey -re
cent_hours 12
```

To get utilization statistics for the last 12 hours using a credential file and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-get-instance-stats.ps1 -aws_credential_file C:\awscreds.conf -re
cent_hours 12
```

To get utilization statistics for the last 12 hours using credentials stored in an environment variable and send the data to Amazon CloudWatch

- Run the following command:

```
.\mon-get-instance-stats.ps1 -recent_hours 12
```

The returned response will be similar to the following example output:

```
Assembly Loaded
Instance Metrics for last 12 hours.
CPU Utilization
Average:  4.69 % Maximum:  10.47 % Minimum:  1.16 %

Memory Utilization
Average:  14.45 % Maximum:  14.77 % Minimum:  14.38 %

pagefileUtilization(c:\pagefile.sys)
Average:  0.00 % Maximum:  0.00 % Minimum:  0.00 %

Volume Utilization C:
Average:  17.28 % Maximum:  17.28 % Minimum:  17.28 %

Volume Utilization D:
Average:  1.41 % Maximum:  1.41 % Minimum:  1.41 %

pagefileUtilization(f:\pagefile.sys)
Average:  0.00 % Maximum:  0.00 % Minimum:  0.00 %
pagefileUtilization(f:\pagefile.sys)
Average:  0  Maximum:  0  Minimum:  0

pagefileUtilization(f:\pagefile.sys)
Average:  0  Maximum:  0  Minimum:  0
```

Set Up Task Scheduler to Send Metrics Reports to Amazon CloudWatch

You can use Windows Task Scheduler to send metrics reports periodically to Amazon CloudWatch.

To set up task scheduler to send metrics reports to Amazon CloudWatch

1. On your Windows Server instance, click **Start**, click **Administrative Tools**, and then click **Task Scheduler**.
2. On the **Action** menu, click **Create Task**.
3. In the **Create Task** dialog box, on the **General** tab, in the **Name** box, type a name for the task, and then select **Run whether user is logged on or not**.
4. On the **Triggers** tab, click **New**.
5. In the **New Trigger** dialog box, under **Settings**, select **One time**.
6. Under **Advanced settings**, select **Repeat task every** and select **5 minutes** from the drop-down menu.
7. In the **for a duration of** drop-down menu, select **Indefinitely**, and then click **OK**.

   Note

   These settings create a trigger that will launch the script every 5 minutes indefinitely. To modify this task to run for set number of days using the **Expire** checkbox.
8. On the **Actions** tab, click **New**.
9. In the **Action** drop-down menu, select **Start a program**.
10. Under **Settings**, in the **Program/script** box, type **Powershell.exe**.
11. In the **Add arguments (optional)** box, type **`-command "C:\scripts\mon-put-metrics-disk.ps1 -disk_drive C:,d -disk_space_util -disk_space_units gigabytes -from_scheduler -logfile C:\mylogfile.log`**, and then click **OK**.
12. On the **Create Task** dialog box, click **OK**.

    If you selected a user account to run this task, Task Scheduler will prompt you for user credentials. Enter the user name and password for the account that will run the task, and then click **OK**.

    Note

    If the PerfMon counters you are using don't require administrator privileges, you can run this task using a system account instead of an administrator account. In the **Create Task** dialog box, on the **General** tab, click **Change User or Group**, and then select a system account.

Controlling User Access to Your AWS Account

Topics

Amazon CloudWatch integrates with AWS Identity and Access Management (AWS IAM) so that you can specify which CloudWatch actions a user in your AWS Account can perform. For example, you could create an AWS IAM policy that gives only certain users in your organization permission to use `GetMetricStatistics`. They could then use the action to retrieve data about your cloud resources.

You can't use AWS IAM to control access to CloudWatch data for specific resources. For example, you can't give a user access to CloudWatch data for only a specific set of instances or a specific LoadBalancer. Permissions granted using IAM cover all the cloud resources you use with CloudWatch.

Important

Using Amazon CloudWatch with AWS IAM doesn't change how you use CloudWatch. There are no changes to CloudWatch actions, and no new CloudWatch actions related to users and access control.

For an example of a policy that covers CloudWatch actions, see Example Policy for CloudWatch (p. 138).

No CloudWatch ARNs

CloudWatch itself has no specific resources for you to control access to. Therefore, there are no CloudWatch ARNs for you to use in an AWS IAM policy. You use * as the resource when writing a policy to control access to CloudWatch actions. For more information about ARNs, go to ARNs in *Using AWS Identity and Access Management*.

CloudWatch Actions

In an AWS IAM policy, you can specify any and all actions that CloudWatch offers. The action name must be prefixed with the lowercase string `cloudwatch:`. For example: `cloudwatch:GetMetricStatistics`, `cloudwatch:ListMetrics`, or `cloudwatch:*` (for all CloudWatch actions). For a list of the actions, go to the Amazon CloudWatch API Reference.

CloudWatch Keys

CloudWatch implements the following policy keys, but no others. For more information about policy keys, go to Condition in *Using AWS Identity and Access Management*.

AWS-Wide Policy Keys

- `aws:CurrentTime` (for date/time conditions)
- `aws:EpochTime` (the date in epoch or UNIX time, for use with date/time conditions)
- `aws:SecureTransport` (Boolean representing whether the request was sent using SSL)
- `aws:SourceIp` (the requester's IP address, for use with IP address conditions)
- `aws:UserAgent` (information about the requester's client application, for use with string conditions)

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine if access is allowed.

For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Example Policy for CloudWatch

This section shows a simple policy for controlling user access to Amazon CloudWatch.

Note

In the future, CloudWatch might add new actions that should logically be included in the following policy, based on the policy's stated goals.

Example

The following sample policy allows a group to retrieve CloudWatch data, but only if the group uses SSL with the request.

```
{
   "Statement":[{
      "Effect":"Allow",
      "Action":["cloudwatch:GetMetricStatistics","cloudwatch:ListMetrics"],
      "Resource":"*",
      "Condition":{
         "Bool":{
            "aws:SecureTransport":"true"
            }
         }
      }
   ]
}
```

For general information about IAM, go to:

- Identity and Access Management (IAM)
- AWS Identity and Access Management Getting Started Guide
- Using AWS Identity and Access Management

Glossary

Access Key ID	An alphanumeric token that uniquely identifies a request sender. This ID is associated with your Secret Access Key.
action	With Amazon CloudWatch, an action is an Amazon SNS topic on Auto Scaling policy that is invoked when an alarm detects a threshold change.
Amazon Elastic Compute Cloud	The Amazon Elastic Compute Cloud (Amazon EC2) is a web service that enables you to launch and manage server instances in Amazon's data centers using APIs or available tools and utilities.
Amazon Machine Image	An Amazon Machine Image (AMI) is an encrypted machine image stored in Amazon Simple Storage Service (Amazon S3). It contains all the information necessary to boot instances of your software.
Amazon Simple Queue Service	Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers.
Availability Zone	Amazon EC2 locations are composed of Regions and Availability Zones. Availability Zones are distinct locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.
AutoScalingGroup	Auto Scaling key term. A capacity group is a representation of a distributed application running on multiple Amazon Elastic Compute Cloud (EC2) instances that are to be automatically scaled. For more information, go to the *Auto Scaling Developer Guide*.
basic statistics	Amazon CloudWatch key term. Basic statistics are those derived from a 5-minute granularity.
detailed statistics	Amazon CloudWatch key term. Detailed statistics are those derived from a 1-minute granularity.
dimension	Amazon CloudWatch key term. A dimension contains additional information about a metric. Dimensions are structured as key-value pairs.
EC2 instance	After an AMI is launched in the cloud, the resulting running system is called an instance or EC2 instance. By default, you can run up to 20 EC2 instances. All instances based on the same AMI start out identical and any information on them is lost when the instances are terminated or fail.
LoadBalancer	Elastic Load Balancing key term. A LoadBalancer is represented by a DNS name and provides the single destination to which all requests intended for your

	application should be directed. For more information, go to the *Elastic Load Balancing Developer Guide*.
metric	Amazon CloudWatch key term. An element of time-series data, specific to a namespace, and having one or more dimensions. CloudWatch metrics are the basis for CloudWatch statistics.
namespace	Amazon CloudWatch key term. A namespace is a conceptual element to which you attach one or more metrics.
period, sampling period	Amazon CloudWatch key term. Statistics are sampled over a period you specify, such as 60 seconds. Alarms invoke actions when a state has changed for a number of consecutive periods.
policy	Auto Scaling policy. You can configure a CloudWatch alarm to invoke an Auto Scaling policy.
Region	A geographical area in which you can launch instances (e.g., US, EU).
scaling activity	Auto Scaling key term. A scaling activity is a long-running process that represents a change to your capacity group. Please see the detailed information on this term in the *Auto Scaling Developer Guide*.
Secret Access Key	A key assigned to you by Amazon Web Services (AWS) when you sign up for an AWS account. Used for request authentication.
statistic	Amazon CloudWatch key term. Please see the detailed information on this term, located at the section called “Statistics” (p. 7).
time-series data	Amazon CloudWatch key term. Data provided to CloudWatch as part of a metric. The time value is assumed to be when the value occurred.
trigger	Auto Scaling key term. A trigger is a concept that combines two AWS features: a CloudWatch alarm (configured to watch a specified CloudWatch metric) and an Auto Scaling policy that describes what should happen when the alarm threshold is crossed. For more information, go to the *Auto Scaling Developer Guide*.
unbounded	Term used in Web Service Definition Language (WSDL), i.e. maxOccurs="unbounded", meaning that the number of potential occurrences is not limited by a set number. Very often used when defining a data type that is a list of other types, such as an unbounded list of integers (element members) or an unbounded list of other complex types that are element/members of the list being defined.
unit	Amazon CloudWatch key term. CloudWatch units include `Seconds`, `Percent`, `Bytes`, `Bits`, `Count`, `Bytes/Second`, `Bits/Second`, `Count/Second`, and `None`. For a complete list, go to the MetricDatum data type in the Amazon CloudWatch API Reference.

Document History

The following table describes the important changes to the Amazon CloudWatch Developer Guide. This documentation is associated with the 2010-08-01 release of Amazon CloudWatch. This guide was last updated on 10 May 2012.

Change	Description	Release Date
New scripts	You can now use the Amazon CloudWatch Monitoring Scripts for Windows to produce and consume Amazon CloudWatch custom metrics. For more information, see Amazon CloudWatch Monitoring Scripts for Windows (p. 128).	19 July 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 Monitor Your Estimated Charges Using Amazon CloudWatch (p. 85).	10 May 2012
New scripts	You can now use the Amazon CloudWatch Monitoring Scripts for Linux to produce and consume Amazon CloudWatch custom metrics. For more information, see Amazon CloudWatch Monitoring Scripts for Linux (p. 122).	24 February 2012
New metrics	You can now access six new Elastic Load Balancing metrics that provide counts of various HTTP response codes. For more information, see Elastic Load Balancing Dimensions and Metrics (p. 62).	19 October 2011
New feature	You can now access metrics from Amazon Simple Notification Service and Amazon Simple Queue Service. For more information, see Amazon SNS Dimensions and Metrics (p. 54) and Amazon SQS Dimensions and Metrics (p. 55).	14 July 2011

Change	Description	Release Date
Restructured Guide	Renamed, merged, and moved sections, including the entire User Scenario section: • *CloudWatch Support for AWS Products* is now CloudWatch Metrics, Namespaces, and Dimensions Reference (p. 36) • *List Available Metrics* is now Listing Available Metrics (p. 25) • *Get Statistics on a Metric* is now Getting Statistics for a Metric (p. 27) • *Create an Alarm that Sends Email* is now Creating CloudWatch Alarms (p. 65)	01 July 2011
New section	Added a section that describes how to use AWS Identity and Access Management (AWS IAM). For more information, see Controlling User Access to Your AWS Account (p. 137).	7 June 2011
New Feature	Added information about using the `PutMetricData` API to publish custom metrics. For more information, see Publishing Custom Metrics (p. 96) or go to the Amazon CloudWatch Getting Started Guide.	10 May 2011
Updated Content	Amazon CloudWatch now retains the history of an alarm for two weeks rather than six weeks. With this change, the retention period for alarms matches the retention period for metrics data.	07 April 2011
New link	This service's endpoint information is now located in the Amazon Web Services General Reference. For more information, go to Regions and Endpoints in Amazon Web Services General Reference.	2 March 2011
Updated Content	Added information about using the AWS Management Console to manage Amazon CloudWatch. For more information, see AWS Management Console (p. 10).	11 Feburary 2011
Updated Content	Added a brief discussion about alarms and Auto Scaling. Specifically, alarms continue to invoke Auto Scaling policy notifications for the duration of a threshold breach rather than only after the initial breach. For more information, see Alarms (p. 9).	19 January 2011
Updated Content	Removed `Minimum` and `Maximum` from the list of valid statistics for the Elastic Load Balancing RequestCount metric. The only valid statistic for RequestCount is `Sum`. For a list of all Elastic Load Balancing metrics, see Elastic Load Balancing Dimensions and Metrics (p. 62).	18 January 2011
New feature	Added ability to send Amazon SNS or Auto Scaling notifications when a metric has crossed a threshold. For more information, see Alarms (p. 9).	02 December 2010

Change	Description	Release Date
New feature	A number of CloudWatch actions now include the MaxRecords and NextToken parameters which enable you to control pages of results to display.	02 December 2010
New feature	This service now integrates with AWS Identity and Access Management (IAM). For more information, go to http://aws.amazon.com/iam and to the Using AWS Identity and Access Management.	02 December 2010

Index