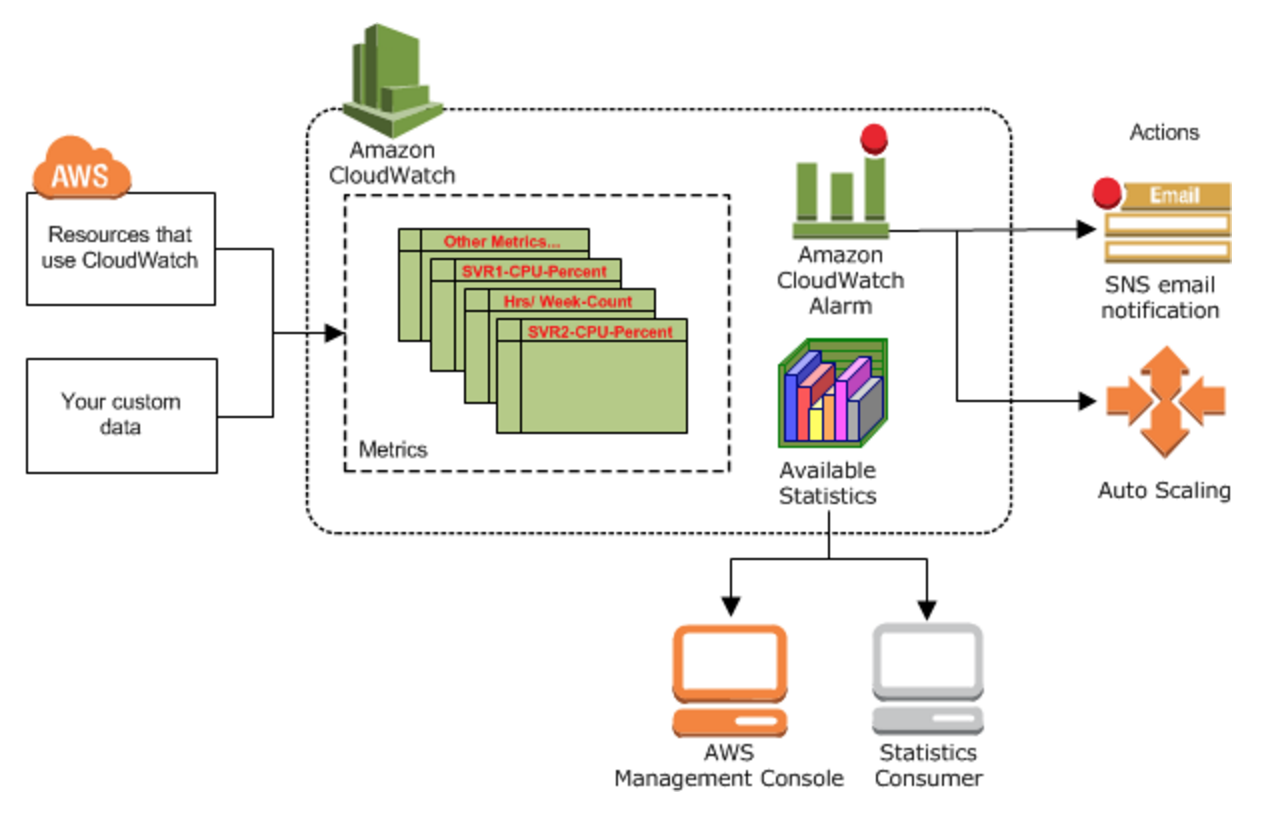
Overview

* real-time monitoring of AWS services & the applications that run on them
* *metrics*: variables you want to measure for your resources and applications
* *alarms*: rules that you define that send notifications or automatically make changes to the resources you are monitoring
* *basic monitoring*: a service sends data points to CloudWatch every five minutes
* *detailed monitoring*: a service sends data points to CloudWatch every minute
* source: <http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_architecture.html>



Concepts

**Metrics**

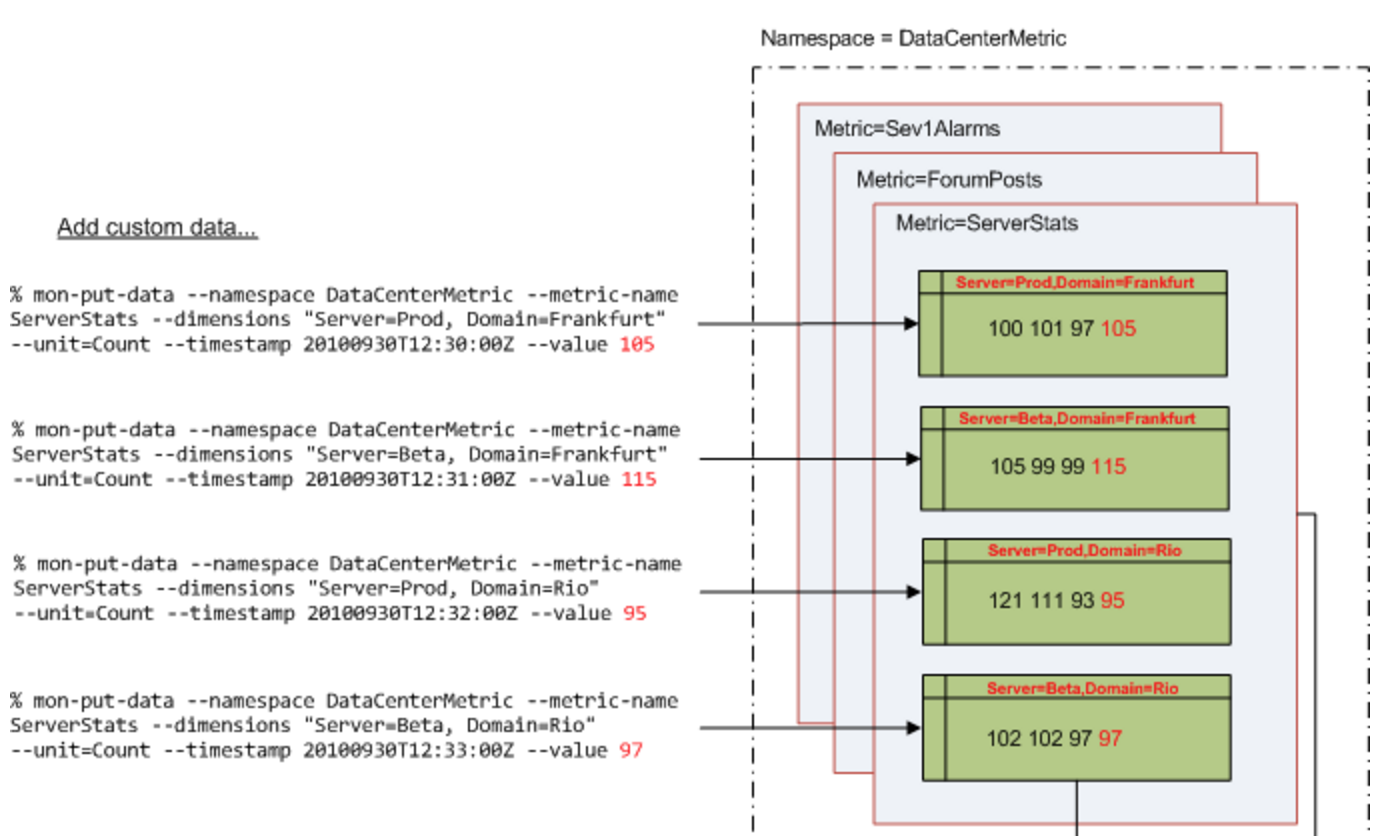
* a time-ordered set of data points that are published to CloudWatch
* can be custom, or AWS-service specific defaults (e.g. EC2 CPU usage consumption)
* you can receive statistics about those data points as an ordered set of time-series data
* uniquely identified by: name, namespace, dimension(s)
* metric data is stored for 2 weeks

**Namespaces**

* 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

**Dimensions**

* name/value pair that helps to identify a metric
* “Every metric has specific characteristics that describe it, and you can think of dimensions as categories for those characteristics”
* CloudWatch treats each unique combination of dimensions as a separate metric
* examples:
  + Server=Prod,Domain=Frankfurt
  + Server=Prod,Domain=Rio



**Units**

* a statistic’s unit of measure
* e.g. Seconds, Bytes, Bits, Percent, Count, Bytes/Second

**Statistics**

* metric data aggregations over specified periods of time
* Aggregations are made using the namespace, metric name, dimensions, and the data point unit of measure, within the time period you specify

**Periods**

* A period is the length of time associated with a specific Amazon CloudWatch statistic
* A period can be as short as one minute (60 seconds) or as long as one day (86,400 seconds)

**Aggregation**

* Amazon CloudWatch doesn’t differentiate the source of a metric…allowing you to [for example] get the statistics for minimum, maximum, average, and sum of all requests across your application

**Alarms**

* Alarms can automatically initiate actions on your behalf, based on parameters you specify
* An alarm watches a single metric over a specified time period, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods

**Regions**

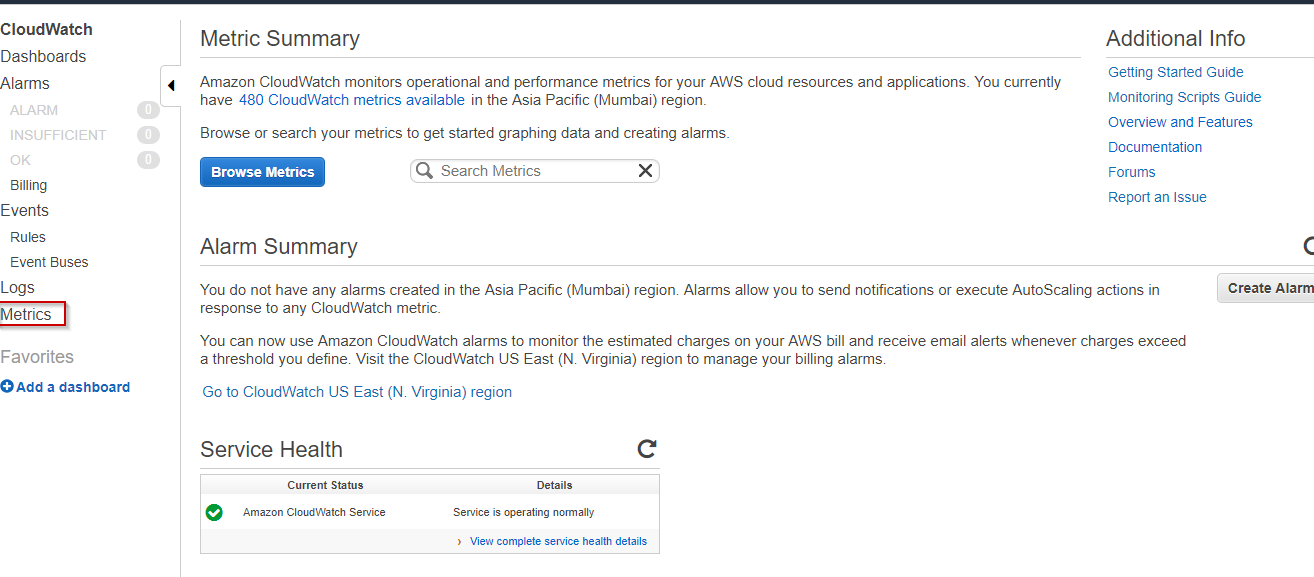
* Amazon CloudWatch does not aggregate data across regions
* metrics are completely separate between regions

**Use Case**: Configure Amazon CloudWatch to send a notification when CPU Utilization of an instance is lower than 15%.

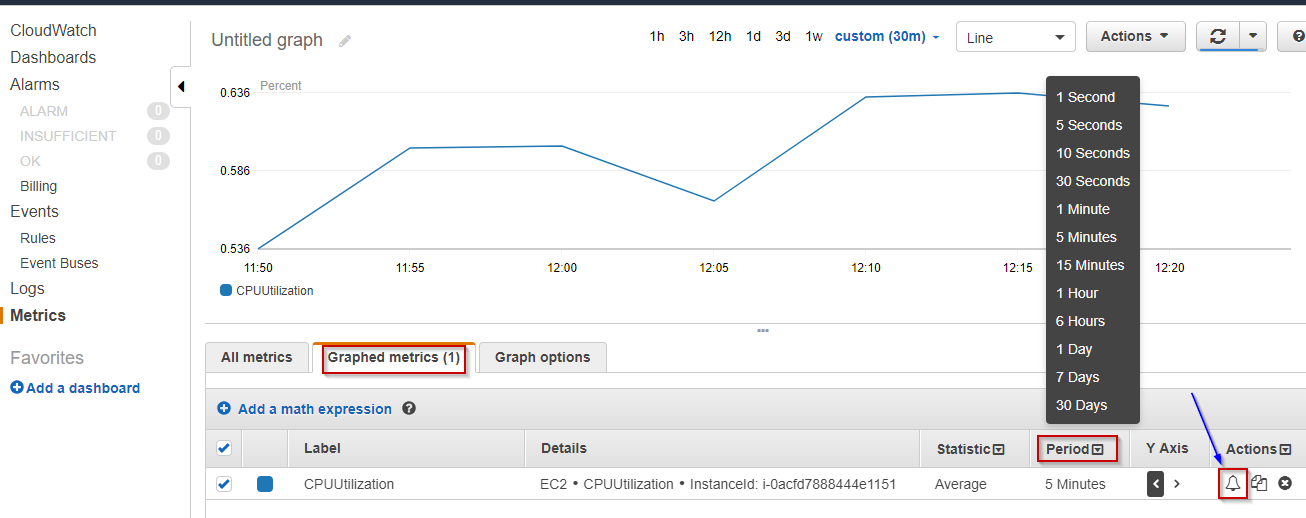
Lets go through various steps involved.

### **Step 1 : Creating a CPU utilization metric**

* Go to Amazon CloudWatch Management Console and select metrics from the navigation pane.

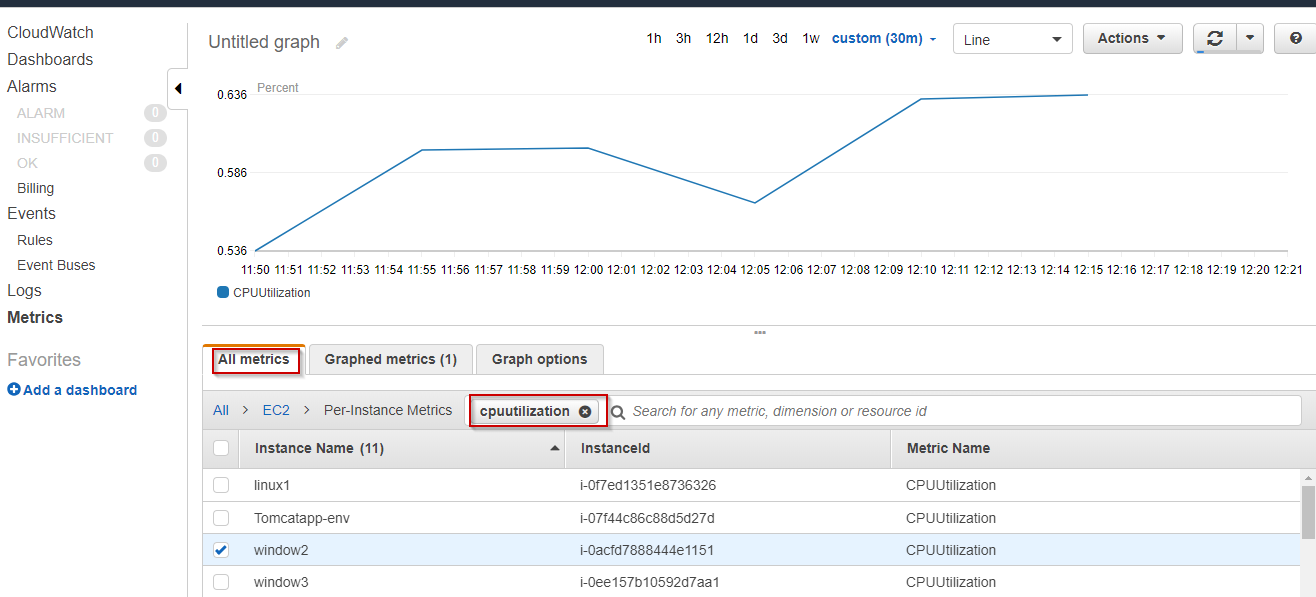


* On the metrics page type CPU Utilization in the search bar.
* From the displayed list of instances choose the instance for which you want to create a metric.

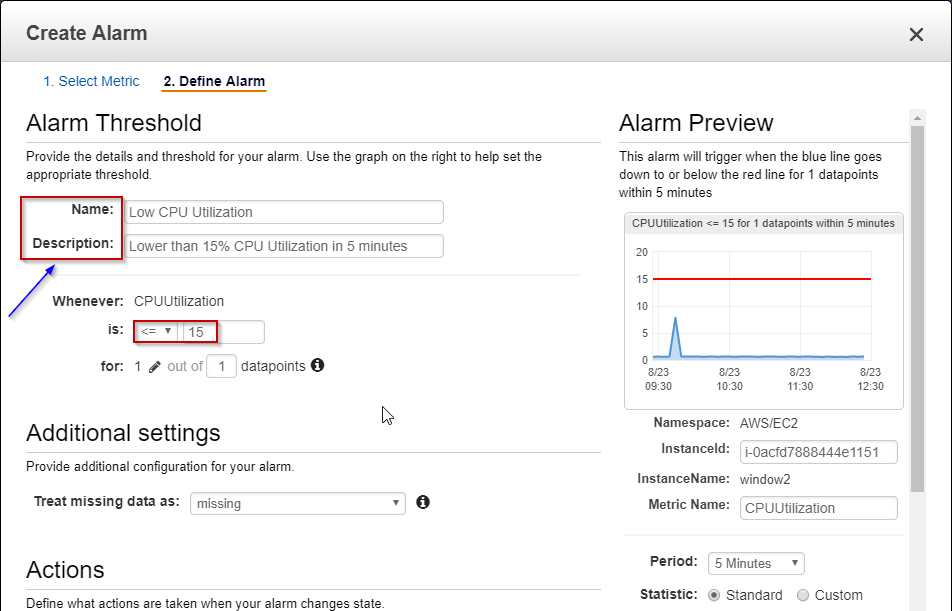


### **Step 2 : Creating an alarm to notify when CPU Utilization metric of the instance is lower than 15%**

* Now select the Graphed Metrics option on the same page. Then set the time period according to your need. And choose an alarm icon located beside the selected instance.

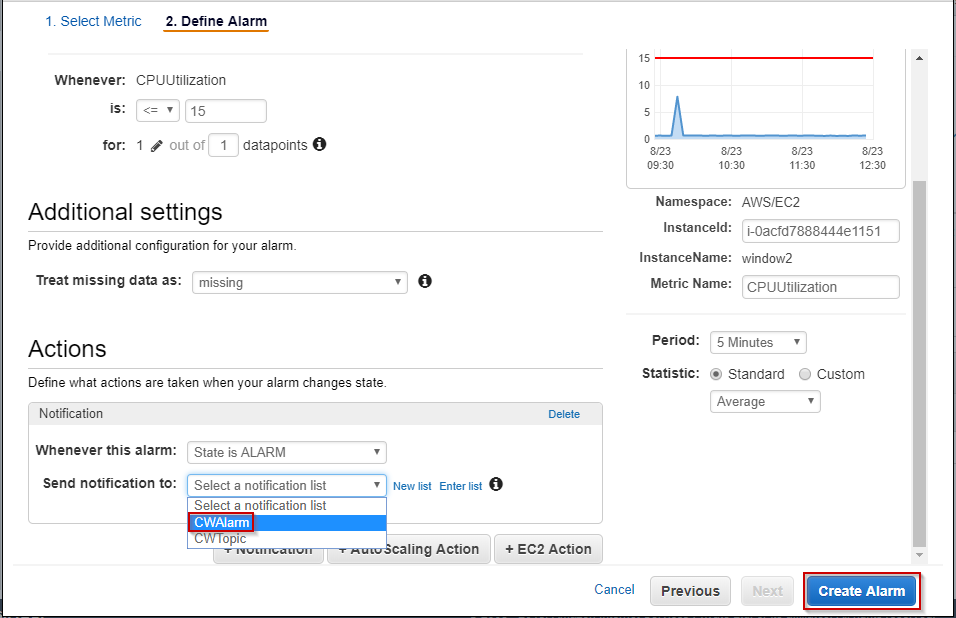


* Configure the alarm in the displayed dialog box. Give your alarm a name and description. Set the Threshold condition.



You want AWS to send you an email notification whenever the alarm condition is satisfied. The notification is sent through Amazon SNS Topic.

* Select New List option if want to add new email recipient, or If you want to choose the existing one, choose Enter List and enter the name of SNS topic.



* Click Create Alarm.

**Congratulations, you have successfully configured Amazon CloudWatch Alarm** to monitor your instance. You will receive the notification through an e-mail on the mail-id you have specified when the alarm condition is met.

Want To Be A Certified AWS Architect? [Enroll Now](https://www.edureka.co/cloudcomputing" \t "_blank)

Now we will talk about the two most important segments of Amazon CloudWatch, which are :

* Amazon CloudWatch Events
* Amazon CloudWatch Logs

## ****Amazon CloudWatch Events****

Amazon CloudWatch Events deliver a real-time stream of system events from AWS resources to AWS Lambda functions, Amazon SNS Topics, Amazon SQS queues, and other target types.

CloudWatch Events enable you to create a set of rules that you can match certain events with. Then you can route these events to one or more targets like Lambda Function, SNS Topic etc. Whenever there are operational changes in your AWS environment, CloudWatch Events capture these changes and perform remedial actions by sending notifications, activating Lambda functions etc.

Let’s talk about certain topics that you need to understand before using CloudWatch Events.

### **Events**

An event indicates a change in the AWS environment. AWS resources generate events when their state changes. Amazon allows you to generate your own custom application-level events and publish them to CloudWatch Events.

### **Rules**

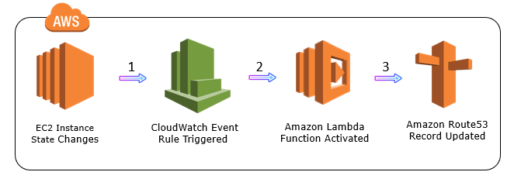
Rules are nothing but constraints. They evaluate every incoming event to determine if out-of-bounds scenario exists. If yes the event is then routed to target for processing. A single rule can route to multiple targets, all of which are processed in parallel.

### **Targets**

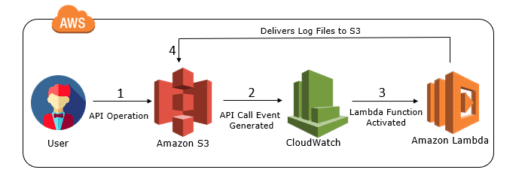
A target processes events. Targets can include Amazon EC2 instances, AWS Lambda functions, Kinesis streams, Amazon ECS tasks, Amazon SNS topics, Amazon SQS queues, and built-in targets. A target receives events in JSON format.

Now let’s have a look at situations where we can use Amazon CloudWatch Events.

**Use Case 1**: You can log the changes in the state of an Amazon EC2 instance by using CloudWatch Events with the assistance of AWS Lambda function.



**Use Case 2**: You can log the object-level API operations on your S3 buckets using CloudWatch Events. But prior to that, you should use AWS CloudTrail to set up a trail configured to receive these operations.



Well, these are just two use-cases which I have specified here so that you will have an idea about the capability of Amazon CloudWatch Events. To describe Amazon CloudWatch Events in one sentence, it is a service that allows you to track changes to your AWS resources with less overhead and more efficiency.

## ****Amazon CloudWatch Logs****

*Amazon CloudWatch Logs is used to monitor, store and access log files from AWS resources like Amazon EC2 instances, Amazon CloudTrail, Route53, and others.*

Let’s take a look at a few basic concepts of Amazon CloudWatch Logs. The below table gives an overview of those concepts.

|  |  |
| --- | --- |
| Log Events | Log Event is a record of some activity recorded by the application or resource being monitored |
| Log Streams | A log stream is a sequence of log events that share the same source. It represents the sequence of events coming from the application instance |
| Log Groups | Log groups represent groups of log streams that share the same retention, monitoring, and access control settings. Each log stream has to belong to one log group. |

With Amazon CloudWatch Logs you can troubleshoot your system errors and maintain and store the respective log files automatically. You can configure an alarm so that a notification will be sent when some error occurs in your system log. You can then troubleshoot the errors within minutes by accessing the original log data stored by CloudWatch Logs. Moreover, you can use Amazon CloudWatch Logs to:

* Store your log data in the highly durable storage
* Monitor your application log files in real-time for specific phrases, values or patterns
* Log information about the DNS queries that Route 53 receives
* Adjust the retention policy for each log group, by choosing a retention period between 10 years and one day.

Now that we have a foundation of Amazon CloudWatch lets go ahead and look at few reasons as to why it is the most famous cloud monitoring tool.

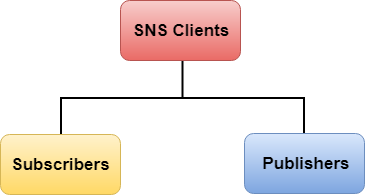
## ****Benefits of Amazon CloudWatch****

* Amazon CloudWatch allows you to access all your data from a single platform. It is natively integrated with more than 70 AWS services. Vodafone company uses Amazon CloudWatch with Auto Scaling groups to monitor CPU usage and to scale from three Amazon EC2 instances to nine during peak periods automatically.
* Provides real-time insights so that you can optimize operational costs and AWS resources. Kellogg company uses Amazon CloudWatch for monitoring, which helps the company make better decisions around the capacity they need, so that they can avoid wastage.
* Provides complete visibility across your applications, infrastructure stack, and AWS services. Atlassian uses Amazon CloudWatch to monitor RAM usage and bandwidth, so they can more easily optimize their application.

# **What is SNS?**

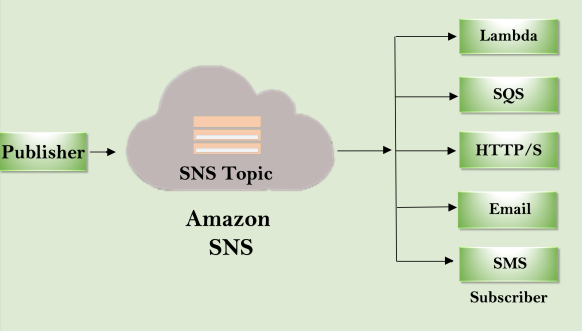
* SNS stands for Simple Notification Service.
* It is a web service which makes it easy to set up, operate, and send a notification from the cloud.
* It provides developers with the highly scalable, cost-effective, and flexible capability to publish messages from an application and sends them to other applications.
* It is a way of sending messages. When you are using AutoScaling, it triggers an SNS service which will email you that "your EC2 instance is growing".
* SNS can also send the messages to devices by sending push notifications to Apple, Google, Fire OS, and Windows devices, as well as Android devices in China with Baidu Cloud Push.
* Besides sending the push notifications to the mobile devices, Amazon SNS sends the notifications through SMS or email to an Amazon Simple Queue Service (SQS), or to an HTTP endpoint.
* SNS notifications can also trigger the Lambda function. When a message is published to an SNS topic that has a Lambda function associated with it, Lambda function is invoked with the payload of the message. Therefore, we can say that the Lambda function is invoked with a message payload as an input parameter and manipulate the information in the message and then sends the message to other SNS topics or other AWS services.
* Amazon SNS allows you to group multiple recipients using topics where the topic is a logical access point that sends the identical copies of the same message to the subscribe recipients.
* Amazon SNS supports multiple endpoint types. For example, you can group together IOS, Android and SMS recipients. Once you publish the message to the topic, SNS delivers the formatted copies of your message to the subscribers.
* To prevent the loss of data, all messages published to SNS are stored redundantly across multiple availability zones.

## SNS Publishers and Subscribers



Amazon SNS is a web service that manages sending messages to the subscribing endpoint. There are two clients of SNS:

* Subscribers
* Publishers



**Publishers**

Publishers are also known as producers that produce and send the message to the SNS which is a logical access point.

**Subscribers**

5.5M

494

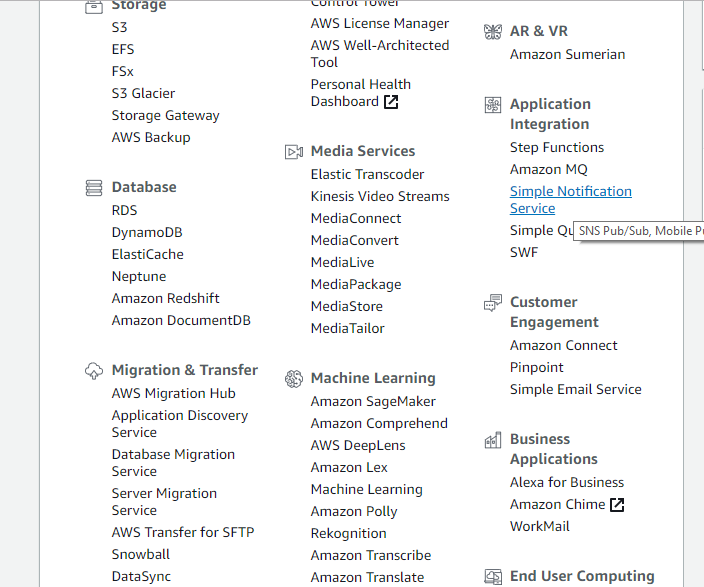
Amazon Web Services Launches Space-Focused Business Unit

Subscribers such as web servers, email addresses, Amazon SQS queues, AWS Lambda functions receive the message or notification from the SNS over one of the supported protocols (Amazon SQS, email, Lambda, HTTP, SMS).

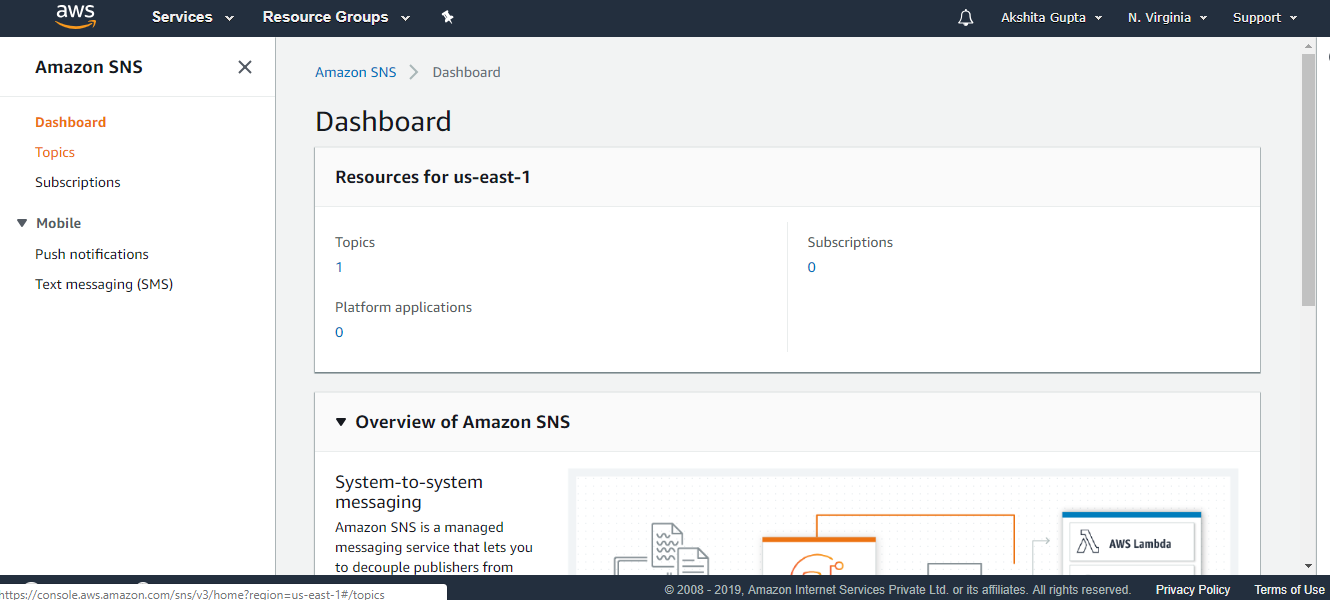
#### **Note: A publisher sends the message to the SNS topic that they have created. There is no need to specify the destination address while publishing the message as the topic itself matches the subscribers associated with the topic that the publisher has created and delivers the message to the subscribers.**

## How to use SNS

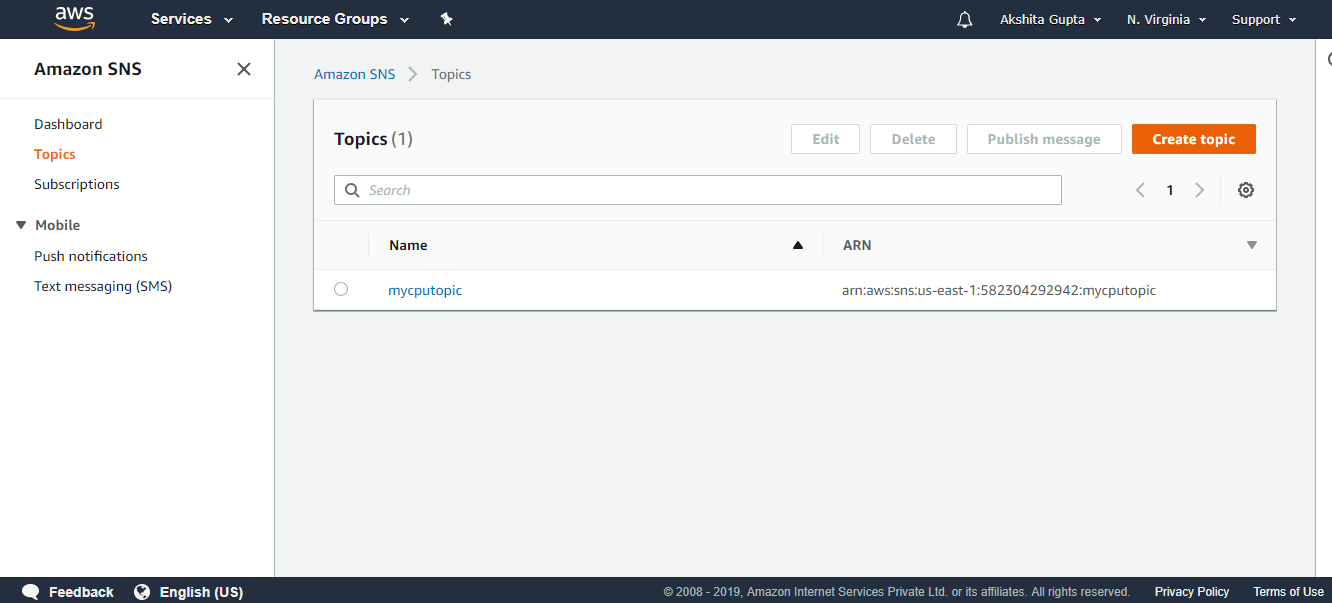
* Move to the SNS service available under the application services.



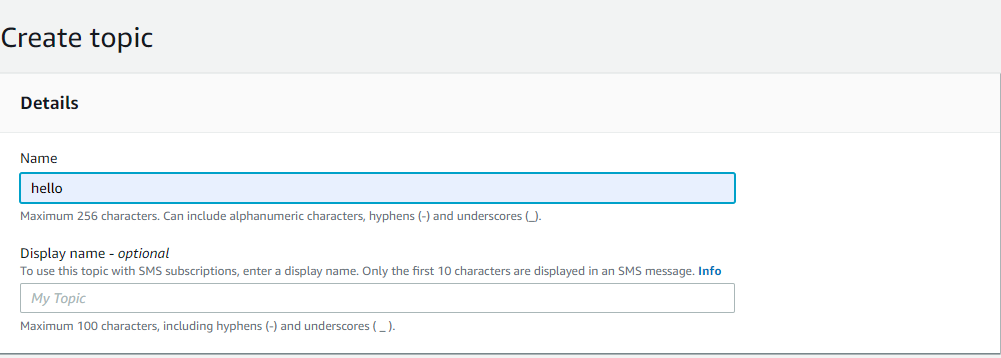
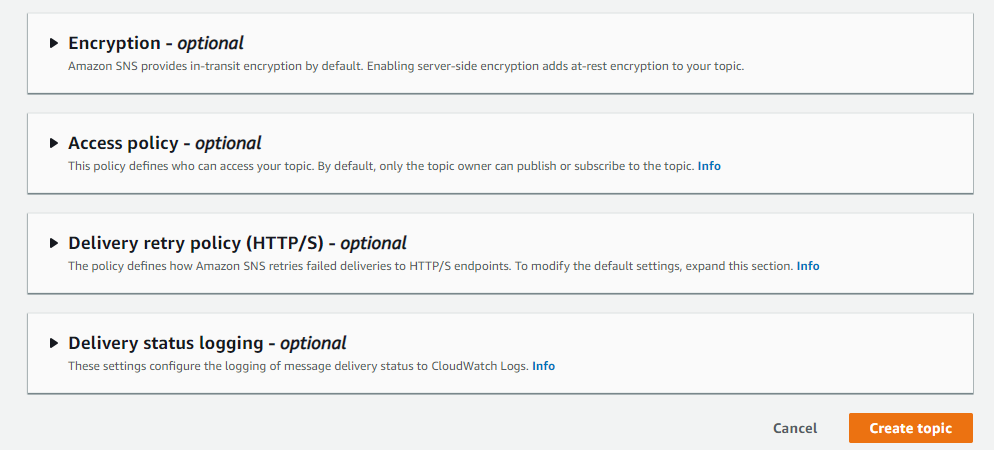
* Click on the **Topics** appearing on the left side of the Console.



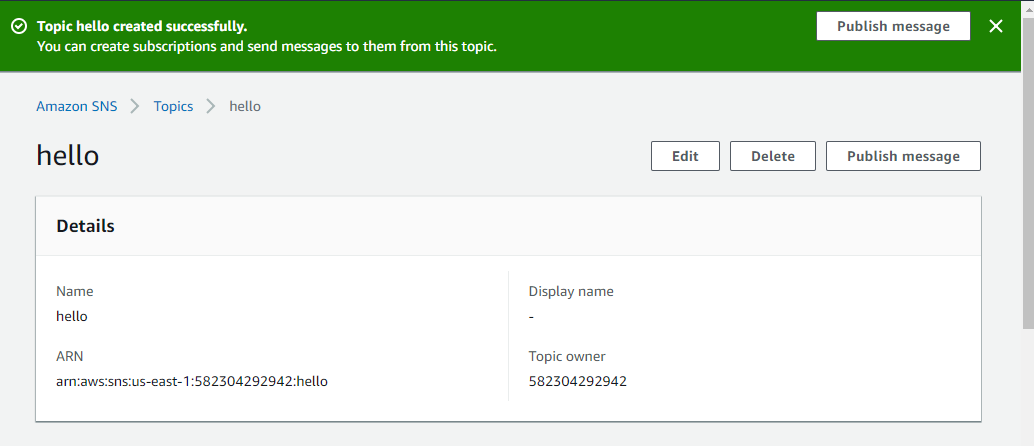
* Click on the **Create Topic** to create a new topic.



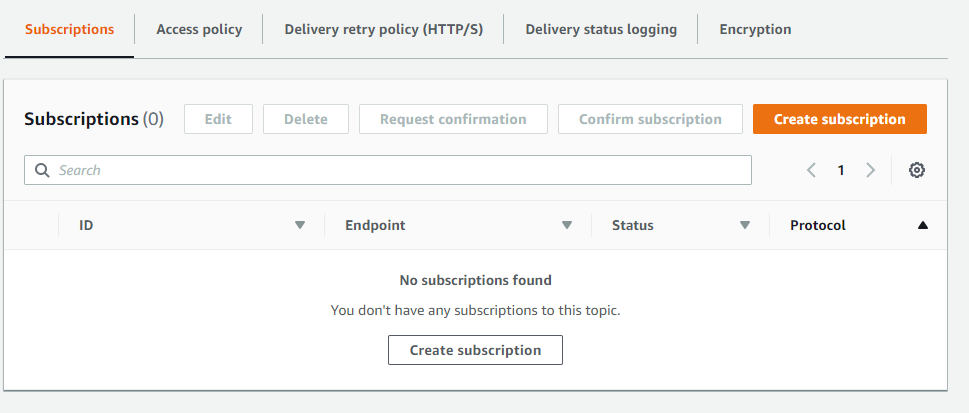
* Enter the topic name in a text box.

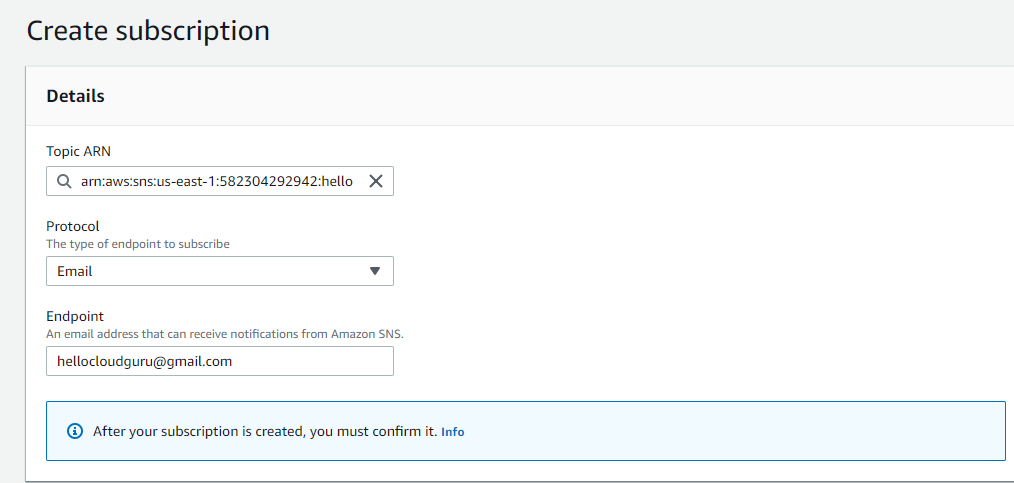
* The below screen shows that the topic has been created successfully.



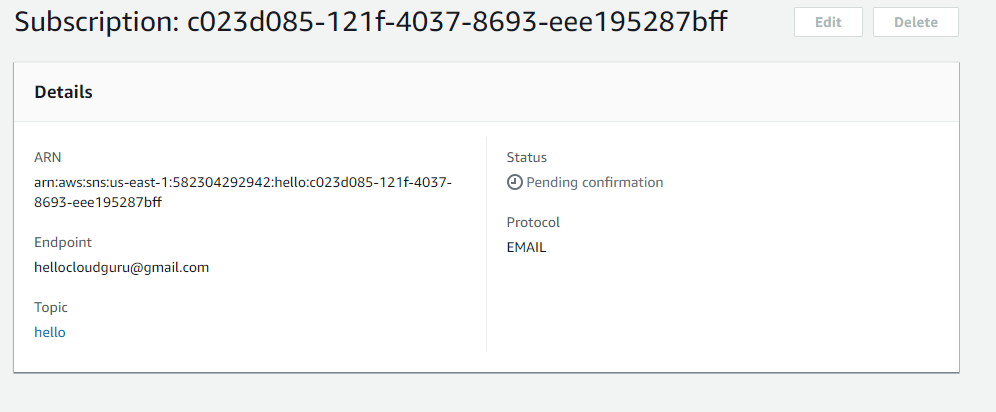
* To create a subscription, click on the **Create subscription**.



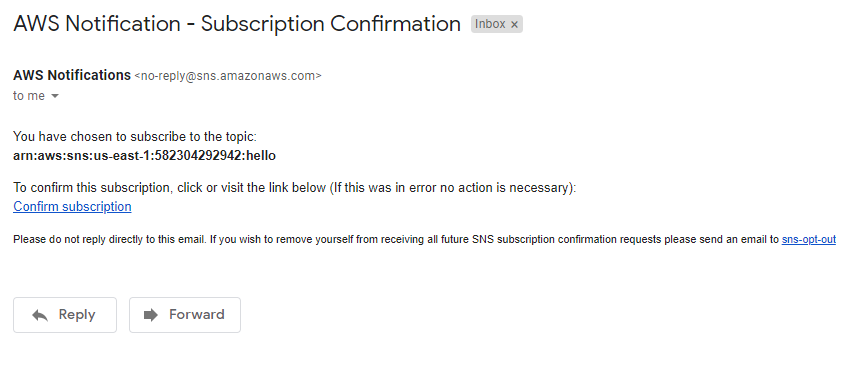
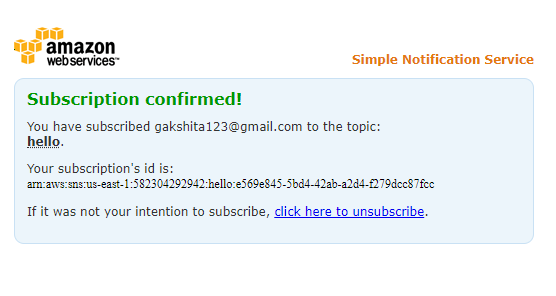
* Now, choose the endpoint type and enter the Endpoint address, i.e., where you want to send your notification.



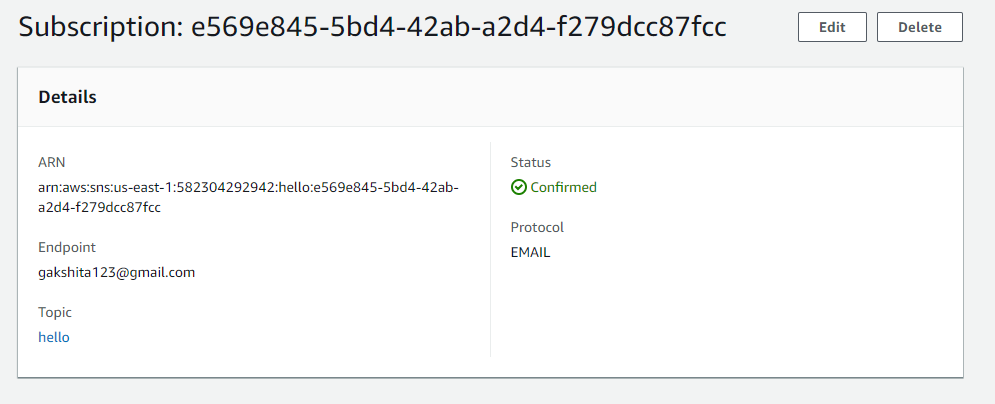
* The below screen shows that the status of subscription is pending.



* The below screen shows that mail has been sent to the subscriber. A Subscriber has to click on the **Confirm Subscription**.

* Click on the topic name, i.e., hello and then click on the **Publish message**.



* Enter the subject, Time to Live and Message body to send to the endpoint.

|  |
| --- |
| SNS SNS   * The message has been sent to all the subscribers that have been mentioned in the ID.   SNS Benefits of SNS SNS   * **Instantaneous delivery** SNS is based on push-based delivery. This is the key difference between SNS and SQS. SNS is pushed once you publish the message in a topic and the message is delivered to multiple subscribers. * **Flexible** SNS supports multiple endpoint types. Multiple endpoint types can receive the message over multiple transport protocols such as email, SMS, Lambda, Amazon SQS, HTTP, etc. * **Inexpensive** SNS service is quite inexpensive as it is based on pay-as-you-go model, i.e., you need to pay only when you are using the resources with no up-front costs. * **Ease of use** SNS service is very simple to use as Web-based AWS Management Console offers the simplicity of the point-and-click interface. * **Simple Architecture** SNS is used to simplify the messaging architecture by offloading the message filtering logic from the subscribers and message routing logic from the publishers. Instead of receiving all the messages from the topic, SNS sends the message to subscriber-only of their interest.  Differences b/w SNS and SQS  * SNS stands for **Simple Notification Service** while SQS stands for **Simple Queue Service**. * SQS is a pull-based delivery, i.e., messages are not pushed to the receivers. Users have to pull the messages from the Queue. SNS is a push-based delivery, i.e., messages are pushed to multiple subscribers. * In SNS service, messages are pushed to the multiple receivers at the same time while in SQS service, messages are not received by the multiple receivers at the same time. * SQS polling introduces some latency in message delivery while SQS pushing pushed the messages to the subscribers immediately. |