Introducing the Amazon Simple Notification Service

We want to make it even easier for developers to build highly functional and architecturally complex applications on AWS. It turns out that applications of this type can often benefit from a publish/subscribe messaging paradigm. In such a system, publishers and receivers of messages are decoupled and unaware of each other's existence. The receivers (also known as subscribers) express interest in certain topics. The senders (publishers) can send a message to a topic. The message will then be immediately delivered or pushed to all of the subscribers to the topic.

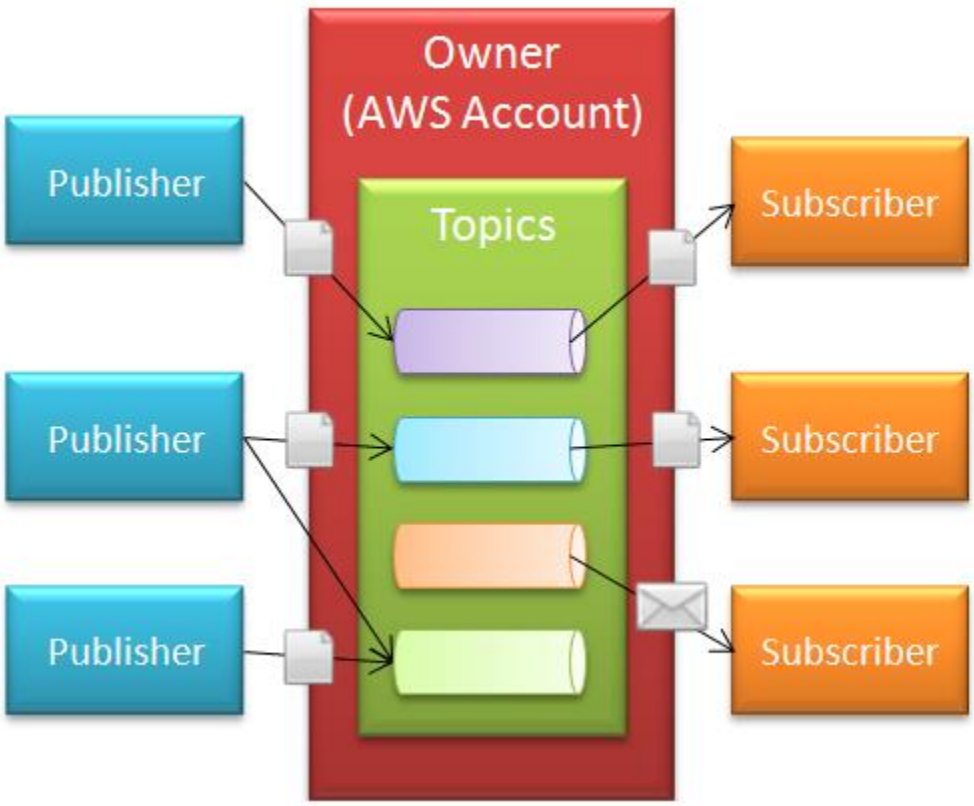
The Amazon Simple Notification Service (SNS) makes it easy for you to build an application in this way. You'll need to know the following terms in order to understand how SNS works:

Topics are named groups of events or acess points, each identifying a specific subject, content, or event type. Each topic has a unique identifier (URI) that identifies the SNS endpoint for publishing and subscribing.

Owners create topics and control all access to the topic. The owner can define the permissions for all of the topics that they own.

Subscribers are clients (applications, end-users, servers, or other devices) that want to receive notifications on specific topics of interest to them.

Publishers send messages to topics. SNS matches the topic with the list of subscribers interested in the topic, and delivers the message to each and every one of them. Here's how it all fits together:



The SNS API is, as the name should imply, clean and simple. Here's what it takes to get started:

1. Call the CreateTopic function to create a new topic. Topic names can be made up of upper and lower case letters, numbers, and hyphens and can be up to 256 characters long.
2. Call the AddPermission function to establish the set of publishers and subscribers with access to the topic.
3. Subscribers call the Subscribe function to express their interest in receiving messages on a particular topic. As part of their request each subscriber must specify a topic, a protocol (HTTP, HTTPS, Email, or Email-JSON), and an endpoint (a URL for HTTP or HTTPS, an email address for either flavor of Email). SNS is also integrated with other AWS services for example, you can have the notifications delivered to an SQS queue. A single subscriber can subscribe to the same topic more than once if desired.
4. As part of the subscription process, SNS will deliver a confirmation message with an embedded token to the endpoint. The subscriber must confirm the subscription by clicking a link in an email or using the ConfirmSubscription function in order to initiate message delivery.
5. Publishers call the Publish function to post messages to a topic which will immediately trigger delivery of the message to each of the topic's subscribers.

Other useful SNS functions include ListTopics (enumerate the list of topics that a specific user can access), RemovePermissions, ListSubscriptions (list the caller's subscriptions), and ListSubscriptionsByTopic (list all of the subscriptions to a topic).

During the beta period, each AWS account will be able to create up to 100 topics. We'll be evaluating this administrative limit as the beta progresses, and we'll do our best to meet requests for increases.

Pricing: There is no charge for the first 100,000 SNS requests per month, the first 100,000 HTTP or HTTPS notifications and the first 1,000 email notifications. As is often the case with new AWS services, we have a long and detailed road map for this service. Here are a few of the items on it:

- AWS Management Console Support – We're building a point-and-click web-based user interface to access and manage SNS.
- Additional Transports – We expect to add additional delivery transport protocols in the future.

I'm looking forward to seeing some cool applications built around SNS! Workflow systems, mobile applications, news and information updates, and the like can all benefit from SNS. Here are some ideas to get you started:

1. Monitoring Alert Notification System – Watch for failure events in a complex system and route information about them to appropriate people based on the type and complexity of the failure. The events could be at the system level (e.g. a process on an EC2 instance is consuming an excessive amount of memory or CPU time) or at the application level (e.g. an application was not able to communicate with an outside data provider).
2. News Distribution – Watch a web site, a Twitter topic, search results, or an RSS feed for changes. Publish the changes to a series of topics (one per news topic) and allow subscribers to choose the topics that they find to be of interest to them.
3. Control EC2 Instances – Subscribe a large array of EC2 instances to a single topic and publish messages to the topic to control the array. This could be system level, with messages used to tell each instance to update its installed software, report on free disk space. Or, it could be application level, with messages to start and stop application processes, change policies, or update reference data tables.