**Application Integration**

**Amazon SQS**

Q: What are the benefits of Amazon SQS over homegrown or packaged message queuing systems?

Amazon SQS provides several advantages over building your own software for managing message queues or using commercial or open-source message queuing systems that require significant up-front time for development and configuration.

These alternatives require ongoing hardware maintenance and system administration resources. The complexity of configuring and managing these systems is compounded by the need for redundant storage of messages that ensures messages are not lost if hardware fails.

In contrast, Amazon SQS requires no administrative overhead and little configuration. Amazon SQS works on a massive scale, processing billions of messages per day. You can scale the amount of traffic you send to Amazon SQS up or down without any configuration. Amazon SQS also provides extremely high message durability, giving you and your stakeholders added confidence.

Q: How is Amazon SQS different from Amazon SNS?

Amazon SNS allows applications to send time-critical messages to multiple subscribers through a “push” mechanism, eliminating the need to periodically check or “poll” for updates. Amazon SQS is a message queue service used by distributed applications to exchange messages through a polling model, and can be used to decouple sending and receiving components.

Q: How is Amazon SQS different from Amazon MQ?

If you're using messaging with existing applications, and want to move your messaging to the cloud quickly and easily, we recommend you consider [Amazon MQ](https://aws.amazon.com/amazon-mq/). It supports industry-standard APIs and protocols so you can switch from any standards-based message broker to Amazon MQ without rewriting the messaging code in your applications. If you are building brand new applications in the cloud, we recommend you consider Amazon SQS and Amazon SNS. Amazon SQS and SNS are lightweight, fully managed message queue and topic services that scale almost infinitely and provide simple, easy-to-use APIs.

Q: Does Amazon SQS provide message ordering?

Yes. FIFO (first-in-first-out) queues preserve the exact order in which messages are sent and received. If you use a FIFO queue, you don't have to place sequencing information in your messages.

Standard queues provide a loose-FIFO capability that attempts to preserve the order of messages. However, because standard queues are designed to be massively scalable using a highly distributed architecture, receiving messages in the exact order they are sent is not guaranteed.

Q: Does Amazon SQS guarantee delivery of messages?

Standard queues provide at-least-once delivery, which means that each message is delivered at least once.

FIFO queues provide [exactly-once processing](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html#FIFO-queues-exactly-once-processing), which means that each message is delivered once and remains available until a consumer processes it and deletes it. Duplicates are not introduced into the queue.

Q: How is Amazon SQS different from Amazon Kinesis Streams?

Amazon SQS offers a reliable, highly-scalable hosted queue for storing messages as they travel between applications or microservices. It moves data between distributed application components and helps you decouple these components. Amazon SQS provides common middleware constructs such as dead-letter queues and poison-pill management. It also provides a generic web services API and can be accessed by any programming language that the AWS SDK supports. Amazon SQS supports both standard and FIFO queues.

Amazon Kinesis Streams allows real-time processing of streaming big data and the ability to read and replay records to multiple Amazon Kinesis Applications. The Amazon Kinesis Client Library (KCL) delivers all records for a given partition key to the same record processor, making it easier to build multiple applications that read from the same Amazon Kinesis stream (for example, to perform counting, aggregation, and filtering).

Q: Can I use Amazon SQS with other AWS services?

Yes. You can make your applications more flexible and scalable by using Amazon SQS with compute services such as Amazon EC2, Amazon EC2 Container Service (Amazon ECS), and AWS Lambda, as well as with storage and database services such as Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.

Q: Can I use [Java Message Service (JMS)](https://en.wikipedia.org/wiki/Java_Message_Service) with Amazon SQS?

Yes. You can take advantage of the scale, low cost, and high availability of Amazon SQS without the worry and high overhead of running your own JMS cluster.

Amazon provides the [Amazon SQS Java Messaging Library](https://github.com/awslabs/amazon-sqs-java-messaging-lib) that implements the JMS 1.1 specification and uses Amazon SQS as the JMS provider.

Q: How does Amazon SQS handle messages that can't be processed?

In Amazon SQS, you can use the API or the console to configure dead letter queues, which are queues that receive messages from other source queues.

If you make a queue into a dead letter queue, it receives messages after a maximum number of processing attempts cannot be completed. You can use dead letter queues to isolate messages that can't be processed for later analysis.

Q: What is Amazon SQS long polling?

Amazon SQS long polling is a way to retrieve messages from your Amazon SQS queues. While the regular short polling returns immediately, even if the message queue being polled is empty, long polling doesn’t return a response until a message arrives in the message queue, or the long poll times out.

Long polling makes it inexpensive to retrieve messages from your Amazon SQS queue as soon as the messages are available. Using long polling might reduce the cost of using SQS, because you can reduce the number of empty receives.

Q: When should I use Amazon SQS long polling, and when should I use Amazon SQS short polling?

In almost all cases, Amazon SQS long polling is preferable to short polling. Long-polling requests let your queue consumers receive messages as soon as they arrive in your queue while reducing the number of empty ReceiveMessageResponse instances returned.

Amazon SQS long polling results in higher performance at reduced cost in the majority of use cases. However, if your application expects an immediate response from a ReceiveMessage call, you might not be able to take advantage of long polling without some modifications to your application.

For example, if your application uses a single thread to poll multiple queues, switching from short polling to long polling will probably not work, because the single thread will wait for the long-poll timeout on any empty queues, delaying the processing of any queues that might contain messages.

In such an application, it is a good practice to use a single thread to process only one queue, allowing the application to take advantage of the benefits that Amazon SQS long polling provides.

Q: What value should I use for my long-poll timeout?

In general, you should use maximum 20 seconds for a long-poll timeout. Because higher long-poll timeout values reduce the number of empty ReceiveMessageResponse instances returned, try to set your long-poll timeout as high as possible.

If the 20-second maximum doesn't work for your application (see the example in the previous question), set a shorter long-poll timeout, as low as 1 second.

All AWS SDKs work with 20-second long polls by default. If you don't use an AWS SDK to access Amazon SQS, or if you configured your AWS SDK to specifically have a shorter timeout, you might need to modify your Amazon SQS client to allow longer requests or to use a shorter long-poll timeout.

Q: How can I secure the messages in my message queues?

Authentication mechanisms ensure that messages stored in Amazon SQS message queues are secured against unauthorized access. You can control who can send messages to a message queue and who can receive messages from a message queue. For additional security, you can build your application to encrypt messages before they are placed in a message queue.

Amazon SQS has its own resource-based permissions system that uses policies written in the same language as [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/) policies: for example, you can use variables, just like in IAM policies. Amazon SQS supports the HTTP over SSL (HTTPS) and Transport Layer Security (TLS) protocols. Most clients can automatically negotiate to use newer versions of TLS without any code or configuration change. Amazon SQS supports versions 1.0, 1.1, and 1.2 of the Transport Layer Security (TLS) protocol in all regions.

Q: What are the benefits of server-side encryption (SSE) for Amazon SQS?

Server-side encryption (SSE) lets you transmit sensitive data in encrypted queues. SSE protects the contents of messages in Amazon SQS queues using keys managed in the [AWS Key Management Service (AWS KMS)](https://aws.amazon.com/kms/). SSE encrypts messages as soon as Amazon SQS receives them. The messages are stored in encrypted form and Amazon SQS decrypts messages only when they are sent to an authorized consumer.

Q: How long can I keep my messages in Amazon SQS message queues?

Longer message retention provides greater flexibility to allow for longer intervals between message production and consumption.

You can configure the Amazon SQS message retention period to a value from 1 minute to 14 days. The default is 4 days. Once the message retention limit is reached, your messages are automatically deleted.

**Amazon SNS**

Q: What is Amazon Simple Notification Service (Amazon SNS)?

Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from [the cloud](https://aws.amazon.com/what-is-cloud-computing/). It provides developers with a highly scalable, flexible, and cost-effective capability to publish messages from an application and immediately deliver them to subscribers or other applications. It is designed to make web-scale computing easier for developers. Amazon SNS follows the “[publish-subscribe” (pub-sub) messaging paradigm](https://aws.amazon.com/pub-sub-messaging/), with notifications being delivered to clients using a “push” mechanism that eliminates the need to periodically check or “poll” for new information and updates. With simple APIs requiring minimal up-front development effort, no maintenance or management overhead and pay-as-you-go pricing, Amazon SNS gives developers an easy mechanism to incorporate a powerful notification system with their applications.

Q: What are the benefits of using Amazon SNS?

Amazon SNS offers several benefits making it a versatile option for building and integrating loosely-coupled, distributed applications:

* Instantaneous, push-based delivery (no polling)
* Simple APIs and easy integration with applications
* Flexible message delivery over multiple transport protocols
* Inexpensive, pay-as-you-go model with no up-front costs
* Web-based AWS Management Console offers the simplicity of a point-and-click interface

Q: What are some example uses for Amazon SNS notifications?

The Amazon SNS service can support a wide variety of needs including event notification, monitoring applications, workflow systems, time-sensitive information updates, mobile applications, and any other application that generates or consumes notifications. For example, Amazon SNS can be used in workflow systems to relay events among distributed computer applications, move data between data stores or update records in business systems. Event updates and notifications concerning validation, approval, inventory changes and shipment status are immediately delivered to relevant system components as well as end-users. A common pattern is to use SNS to publish messages to [Amazon SQS](https://aws.amazon.com/sqs/) message queues to reliably send messages to one or many system components asynchronously. Another example use for Amazon SNS is to relay time-critical events to mobile applications and devices. Since Amazon SNS is both highly reliable and scalable, it provides significant advantages to developers who build applications that rely on real-time events.

Q: How does Amazon SNS work?

It is very easy to get started with Amazon SNS. Developers must first create a “topic” which is an “access point” – identifying a specific subject or event type – for publishing messages and allowing clients to subscribe for notifications. Once a topic is created, the topic owner can set policies for it such as limiting who can publish messages or subscribe to notifications, or specifying which notification protocols will be supported (i.e. HTTP/HTTPS, email, SMS). Subscribers are clients interested in receiving notifications from topics of interest; they can subscribe to a topic or be subscribed by the topic owner. Subscribers specify the protocol and end-point (URL, email address, etc.) for notifications to be delivered. When publishers have information or updates to notify their subscribers about, they can publish a message to the topic – which immediately triggers Amazon SNS to deliver the message to all applicable subscribers.

Q: How is Amazon SNS different from Amazon SQS?

Amazon Simple Queue Service (SQS) and Amazon SNS are both messaging services within AWS, which provide different benefits for developers. Amazon SNS allows applications to send time-critical messages to multiple subscribers through a “push” mechanism, eliminating the need to periodically check or “poll” for updates. Amazon SQS is a [message queue service](https://aws.amazon.com/message-queue/) used by distributed applications to exchange messages through a polling model, and can be used to decouple sending and receiving components. Amazon SQS provides flexibility for distributed components of applications to send and receive messages without requiring each component to be concurrently available.

A common pattern is to use SNS to publish messages to Amazon SQS queues to reliably send messages to one or many system components asynchronously.

Q: How is Amazon SNS different from Amazon MQ?

Amazon MQ, Amazon SQS, and Amazon SNS are messaging services that are suitable for anyone from startups to enterprises. If you're using messaging with existing applications, and want to move your messaging to the cloud quickly and easily, we recommend you consider [Amazon MQ](https://aws.amazon.com/amazon-mq/). It supports industry-standard APIs and protocols so you can switch from any standards-based message broker to Amazon MQ without rewriting the messaging code in your applications. If you are building brand new applications in the cloud, we recommend you consider Amazon SQS and Amazon SNS. Amazon SQS and SNS are lightweight, fully managed message queue and topic services that scale almost infinitely and provide simple, easy-to-use APIs. You can use Amazon SQS and SNS to decouple and scale microservices, distributed systems, and serverless applications, and improve reliability.

Q: Is Amazon SNS supported in the AWS Management Console?

Amazon SNS is supported in the [AWS Management Console](http://aws.amazon.com/console) which provides a point-and-click, web-based interface to access and manage Amazon SNS. Using the AWS Management Console, you can create topics, add subscribers, and send notifications – all from your browser. In addition, the AWS Management Console makes it easy to publish messages to your endpoint of choice (HTTP, SQS, Lambda, mobile push, email, or SMS) and edit topic policies to control publisher and subscriber access.

Q: Can I get a history of SNS API calls made on my account for security analysis and operational troubleshooting purposes?

Yes. SNS supports [AWS CloudTrail](https://aws.amazon.com/cloudtrail/), a web service that records AWS API calls for your account and delivers log files to you. With CloudTrail, you can obtain a history of such information as the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by SNS.

SNS currently supports CloudTrail auditing for authenticated calls only. CloudTrail Audit logs for unauthenticated ConfirmSubscription and Unsubscribe calls are not available at this time.

To receive a history of SNS API calls made on your account, simply [turn on AWS CloudTrail in the AWS Management Console](https://console.aws.amazon.com/cloudtrail).

Q: What are the different delivery formats/transports for receiving notifications?

In order for customers to have broad flexibility of delivery mechanisms, Amazon SNS supports notifications over multiple transport protocols. Customers can select one the following transports as part of the subscription requests:

* “HTTP”, “HTTPS” – Subscribers specify a URL as part of the subscription registration; notifications will be delivered through an HTTP POST to the specified URL.
* ”Email”, “Email-JSON” – Messages are sent to registered addresses as email. Email-JSON sends notifications as a JSON object, while Email sends text-based email.
* “SQS” – Users can specify an SQS standard queue as the endpoint; Amazon SNS will enqueue a notification message to the specified queue (which subscribers can then process using SQS APIs such as ReceiveMessage, DeleteMessage, etc.). Note that FIFO queues are not currently supported. You can use SNS to forward messages to standard queues.
* “SMS” – Messages are sent to registered phone numbers as SMS text messages.

Q: Can a single topic support subscriptions over multiple protocols/transports?

Subscribers to an Amazon SNS topic can receive notifications on any transport supported by the topic. A topic can support subscriptions and notification deliveries over multiple transports.

Q: Can Amazon SNS be used with other AWS services?

Amazon SNS can be used with other AWS services such as Amazon SQS, Amazon EC2 and Amazon S3. Here is an example of how an order processing workflow system uses Amazon SNS with Amazon EC2, SQS, and SimpleDB. In this workflow system, messages are sent between application components whenever a transaction occurs or an order advances through the order processing pipeline. When a customer initially places an order, the transaction is first recorded in Amazon SimpleDB and an application running on Amazon EC2 forwards the order request to a payment processor which debits the customer’s credit card or bank account. Once approved, an order confirmation message is published to an Amazon SNS topic. In this case, the topic has various subscribers over Email/HTTP – merchant, customer and supply chain partners – and notifications sent by Amazon SNS for that topic can instantly update all of them that payment processing was successful. Notifications can also be used to orchestrate an order processing system running on EC2, where notifications sent over HTTP can trigger real-time processing in related components such as an inventory system or a shipping service. By integrating Amazon SNS with Amazon SQS, all notifications delivered are also persisted in an Amazon SQS queue where they are processed by an auditing application at a future time.

Q: How can users secure the messages sent to my topics?

All API calls made to Amazon SNS are validated for the user’s AWS Id and the signature. In addition, we recommend that users secure their data over the wire by connecting to our secure SSL end-points.

Q: Who can create a topic?

Topics can only be created by users with valid AWS IDs who have signed up for Amazon SNS. The easiest way to create a topic is to use the AWS Management Console. It can also be created through the CreateTopic API.

Q: How durable is my data once published to Amazon SNS?

SNS provides durable storage of all messages that it receives. Upon receiving a publish request, SNS stores multiple copies (to disk) of the message across multiple Availability Zones before acknowledging receipt of the request to the sender. Each AWS Region has multiple, isolated locations known as [Availability Zones](https://aws.amazon.com/about-aws/global-infrastructure/). Although rare, should a failure occur in one zone, the operation of SNS and the durability of your messages continue without disruption.

Q: Does Amazon SNS guarantee that messages are delivered to the subscribed endpoint?

When a message is published to a topic, Amazon SNS will attempt to deliver notifications to all subscribers registered for that topic. Due to potential Internet issues or Email delivery restrictions, sometimes the notification may not successfully reach an HTTP or Email end-point. In the case of HTTP, an SNS Delivery Policy can be used to control the retry pattern (linear, geometric, exponential backoff), maximum and minimum retry delays, and other parameters. If it is critical that all published messages be successfully processed, developers should have notifications delivered to an SQS queue (in addition to notifications over other transports).

Q: Are there limits to the number of topics or number of subscribers per topic?

By default, SNS offers 10 million subscriptions per topic, and 100,000 topics per account.

Q: What is direct addressing? How does it work?

Direct addressing allows you to deliver notifications directly to a single endpoint, rather than sending identical messages to all subscribers of a topic. This is useful if you want to deliver precisely targeted messages to each recipient. When you register device tokens with SNS, SNS creates an endpoint that corresponds to the token. You can publish to the token endpoint just as you would publish to a topic.

You can direct publish either the text of your notification, or a platform-specific payload that takes advantage of platform-specific features such as updating the badge count of your app. Direct addressing is currently only available for push notifications endpoints.

Q: Does SNS support direct addressing for SMS or Email?

At this time, direct addressing is only supported for mobile push endpoints (APNS, FCM, ADM, WNS, MPNS, Baidu) and SMS. Email messaging requires the use of topics.

Q: Can I monitor my push notifications through Amazon CloudWatch?

Yes. SNS publishes Cloudwatch metrics for number of messages published, number of successful notifications, number of failed notifications, number of notifications filtered out, and size of data published. Metrics are available on per application basis. You can access Cloudwatch metrics via AWS Management Console or CloudWatch APIs.

Q: What is Time to Live (TTL)?  
  
Some messages that you can send with SNS are relevant or valuable only for a limited period of time. Amazon SNS now allows you to set a TTL (Time to Live) value for each message. When the TTL expires for a given message that was not delivered and read by an end user, the message is deleted. TTL is specified in seconds and is relative to the time Publish call is made.

Q: What does support for AWS Lambda endpoints in Amazon SNS mean?

You can invoke your AWS Lambda functions by publishing messages to Amazon SNS topics that have AWS Lambda functions subscribed to them. Because Amazon SNS supports message fan-out, publishing a single message can invoke different AWS Lambda functions or invoke Lambda functions in addition to delivering notifications to supported Amazon SNS destinations such as mobile push, HTTP endpoints, SQS, email and SMS.

Q: What can I do with AWS Lambda functions and Amazon SNS?

By subscribing AWS Lambda functions to Amazon SNS topics, you can perform custom message handling. You can invoke an AWS Lambda function to provide custom message delivery handling by first publishing a message to an AWS Lambda function, have your Lambda function modify a message (e.g. localize language) and then filter and route those messages to other topics and endpoints. Apps and services that already send Amazon SNS notifications, such as Amazon CloudWatch, can now immediately take advantage of AWS Lambda without having to provision or manage infrastructure for custom message handling. You can also use delivery to an AWS Lambda function as a way to publish to other AWS services such as Amazon Kinesis or Amazon S3. You can subscribe an AWS Lambda function to the Amazon SNS topic, and then have the Lambda function in turn write to another service.

Q: How do I activate AWS Lambda endpoint support in Amazon SNS?

You need to first create an AWS Lambda function via your AWS account and the [AWS Lambda console](https://console.aws.amazon.com/lambda), and then subscribe that AWS Lambda function to a topic using the [Amazon SNS console](https://us-west-2.console.aws.amazon.com/sns/v2/home) or the [Amazon SNS APIs](http://docs.aws.amazon.com/sns/latest/dg/mobile-push-api.html). Once that is complete, any messages that you publish to the Amazon SNS topics which have Lambda functions subscribed to them will be delivered to the appropriate Lambda functions in addition to any other destinations subscribed to that topic.

**Amazon SWF**

Q: What is Amazon SWF?

Amazon Simple Workflow Service (SWF) is a web service that makes it easy to coordinate work across distributed application components. Amazon SWF enables applications for a range of use cases, including media processing, web application back-ends, business process workflows, and analytics pipelines, to be designed as a coordination of tasks. Tasks represent invocations of various processing steps in an application which can be performed by executable code, web service calls, human actions, and scripts.

The coordination of tasks involves managing execution dependencies, scheduling, and concurrency in accordance with the logical flow of the application. With Amazon SWF, developers get full control over implementing processing steps and coordinating the tasks that drive them, without worrying about underlying complexities such as tracking their progress and keeping their state. Amazon SWF also provides the AWS Flow Framework to help developers use asynchronous programming in the development of their applications. By using Amazon SWF, developers benefit from ease of programming and have the ability to improve their applications’ resource usage, latencies, and throughputs.

Q: What are the benefits of designing my application as a coordination of tasks? How does Amazon SWF help me with this?

In Amazon SWF, tasks represent invocations of logical steps in applications. Tasks are processed by workers which are programs that interact with Amazon SWF to get tasks, process them, and return their results. A worker implements an application processing step. You can build workers in different programming languages and even reuse existing components to quickly create the worker. For example, you can use cloud services, enterprise applications, legacy systems, and even simple scripts to implement workers. By independently controlling the number of workers for processing each type of task, you can control the throughput of your application efficiently.

To coordinate the application execution across workers, you write a program called the decider in your choice of programming language. The separation of processing steps and their coordination makes it possible to manage your application in a controlled manner and give you the flexibility to deploy, run, scale and update them independently. You can choose to deploy workers and deciders either in [the cloud](https://aws.amazon.com/what-is-cloud-computing/) (e.g. Amazon EC2 or Lambda) or on machines behind corporate firewalls. Because of the decoupling of workers and deciders, your business logic can be dynamic and you application can be quickly updated to accommodate new requirements. For example, you can remove, skip, or retry tasks and create new application flows simply by changing the decider.

By implementing workers and deciders, you focus on your differentiated application logic as it pertains to performing the actual processing steps and coordinating them. Amazon SWF handles the underlying details such as storing tasks until they can be assigned, monitoring assigned tasks, and providing consistent information on their completion. Amazon SWF also provides ongoing visibility at the level of each task through APIs and a console.

Q: What can I do with Amazon SWF?

Amazon SWF can be used to address many challenges that arise while building applications with distributed components. For example, you can use Amazon SWF and the accompanying AWS Flow Framework for:

* Writing your applications as asynchronous programs using simple programming constructs that abstract details such as initiating tasks to run remotely and tracking the program’s runtime state.
* Maintaining your application’s execution state (e.g. which steps have completed, which ones are running, etc.). You do not have to use databases, custom systems, or ad hoc solutions to keep execution state.
* Communicating and managing the flow of work between your application components. With Amazon SWF, you do not need to design a messaging protocol or worry about lost and duplicated tasks.
* Centralizing the coordination of steps in your application. Your coordination logic does not have to be scattered across different components, but can be encapsulated in a single program.
* Integrating a range of programs and components, including legacy systems and 3rd party cloud services, into your applications. By allowing your application flexibility in where and in what combination the application components are deployed, Amazon SWF helps you gradually migrate application components from private data centers to public cloud infrastructure without disrupting the application availability or performance.
* Automating workflows that include long-running human tasks (e.g. approvals, reviews, investigations, etc.) Amazon SWF reliably tracks the status of processing steps that run up to several days or months.
* Building an application layer on top of Amazon SWF to support domain specific languages for your end users. Since Amazon SWF gives you full flexibility in choosing your programming language, you can conveniently build interpreters for specialized languages (e.g. XPDL) and customized user-interfaces including modeling tools.
* Getting detailed audit trails and visibility into all running instances of your applications. You can also incorporate visibility capabilities provided by Amazon SWF into your own user interfaces using the APIs provided by Amazon SWF.

Customers have used Amazon SWF to build applications for video encoding, social commerce, infrastructure provisioning, MapReduce pipelines, business process management, and several other use cases. For more details on use cases, please see What are some use cases that can be solved with SWF?. To see how customers are using Amazon SWF today, please read our case studies.

Q: What are the benefits of Amazon SWF vs. homegrown solutions and existing workflow products?

When building solutions to coordinate tasks in a distributed environment, developers have to account for several variables. Tasks that drive processing steps can be long-running and may fail, timeout, or require restarts. They often complete with varying throughputs and latencies. Tracking and visualizing tasks in all these cases is not only challenging, but is also undifferentiated work. As applications and tasks scale up, developers face difficult distributed systems’ problems. For example, they must ensure that a task is assigned only once and that its outcome is tracked reliably through unexpected failures and outages. By using Amazon SWF, developers can focus on their differentiated application logic, i.e. how to process tasks and how to coordinate them.

Existing workflow products often force developers to learn specialized languages, host expensive databases, and give up control over task execution. The specialized languages make it difficult to express complex applications and are not flexible enough for effecting changes quickly. Amazon SWF, on the other hand, is a cloud-based service, allows common programming languages to be used, and lets developers control where tasks are processed. By adopting a loosely coupled model for distributed applications, Amazon SWF enables changes to be made in an agile manner.

Q: What are some use cases that can be solved with Amazon SWF?

Amazon SWF has been applied to use cases in media processing, business process automation, data analytics, migration to the cloud, and batch processing. Some examples are:

Use case #1: Video encoding using Amazon S3 and Amazon EC2. In this use case, large videos are uploaded to Amazon S3 in chunks. The upload of chunks has to be monitored. After a chunk is uploaded, it is encoded by downloading it to an Amazon EC2 instance. The encoded chunk is stored to another Amazon S3 location. After all of the chunks have been encoded in this manner, they are combined into a complete encoded file which is stored back in its entirety to Amazon S3. Failures could occur during this process due to one or more chunks encountering encoding errors. Such failures need to be detected and handled.

With Amazon SWF: The entire application is built as a workflow where each video file is handled as one workflow execution. The tasks that are processed by different workers are: upload a chunk to Amazon S3, download a chunk from Amazon S3 to an Amazon EC2 instance and encode it, store a chunk back to Amazon S3, combine multiple chunks into a single file, and upload a complete file to Amazon S3. The decider initiates concurrent tasks to exploit the parallelism in the use case. It initiates a task to encode an uploaded chunk without waiting for other chunks to be uploaded. If a task for a chunk fails, the decider re-runs it for that chunk only. The application state kept by Amazon SWF helps the decider control the workflow. For example, the decider uses it to detect when all chunks have been encoded and to extract their Amazon S3 locations so that they can be combined. The execution’s progress is continuously tracked in the Amazon SWF Management Console. If there are failures, the specific tasks that failed are identified and used to pinpoint the failed chunks.

Use case #2: Migrating components from the datacenter to the cloud. Business critical operations are hosted in a private datacenter but need to be moved entirely to the cloud without causing disruptions.

With Amazon SWF: Amazon SWF-based applications can combine workers that wrap components running in the datacenter with workers that run in the cloud. To transition a datacenter worker seamlessly, new workers of the same type are first deployed in the cloud. The workers in the datacenter continue to run as usual, along with the new cloud-based workers. The cloud-based workers are tested and validated by routing a portion of the load through them. During this testing, the application is not disrupted because the workers in the datacenter continue to run. After successful testing, the workers in the datacenter are gradually stopped and those in the cloud are scaled up, so that the workers are eventually run entirely in the cloud. This process can be repeated for all other workers in the datacenter so that the application moves entirely to the cloud. If for some business reason, certain processing steps must continue to be performed in the private data center, those workers can continue to run in the private data center and still participate in the application.

Q: I run a large number of mission critical application executions. How can I monitor and scale them?

In addition to a Management Console, Amazon SWF provides a comprehensive set of visibility APIs. You can use these to get run-time information to monitor all your executions and to auto-scale your executions depending on load. You can get detailed data on each workflow type, such as the count of open and closed executions in a specified time range. Using the visibility APIs, you can also build your own custom monitoring applications.