
aws

Why data pipelines on cloud are cheaper than on-prem solution?

Data pipelines on the cloud are often cheaper than on-premises solutions due to several reasons:

- **Infrastructure Costs**: Cloud platforms eliminate the need to invest in physical hardware, reducing both upfront capital expenses and ongoing maintenance costs.
- **Scalability**: Cloud services allow for on-demand scalability. Instead of over-provisioning resources, businesses can scale up or down based on actual usage, optimizing costs.
- **Operational Efficiency**: Cloud providers handle much of the routine maintenance, patching, and updates, reducing the manpower required for these tasks on-premises.
- **Resource Utilization**: Cloud platforms typically offer better resource utilization and multi-tenancy benefits, allowing users to only pay for the resources they consume.
- **Reduced Overhead**: On-premises solutions often require dedicated IT teams for setup, maintenance, and troubleshooting. Cloud solutions can reduce or eliminate these overhead costs.
- **Flexibility**: The cloud offers various pricing models, such as pay-as-you-go, reserved instances, or spot instances, enabling cost optimizations based on workload patterns.
- **Innovation Pace**: Cloud providers continuously introduce new features, tools, and integrations, often at no additional cost. This can lead to more efficient and cost-effective pipeline designs over time.
- **Disaster Recovery & Redundancy**: Implementing redundancy and disaster recovery on-premises can be expensive. Cloud providers often offer these features at a fraction of the cost.

Common Services by AWS - Azure - GCP

Grow Data Skills

	Azure	AWS	Google
Available Regions	Azure Regions	AWS Regions and Zones	Google Compute Regions & Zones
Compute Services	Virtual Machines	Elastic Compute Cloud (EC2)	Compute Engine
App Hosting	Azure Cloud Services	Amazon Elastic Beanstalk	Google App Engine
Serverless Computing	Azure Functions	AWS Lambda	Google Cloud Functions
Container Support	Azure Container Service	EC2 Container Service	Container Engine
Scaling Options	Azure Autoscale	Auto Scaling	Autoscaler
Object Storage	Azure Blob Storage	Amazon Simple Storage (S3)	Cloud Storage
Block Storage	Azure Managed Storage	Amazon Elastic Block Storage	Persistent Disk
Content Delivery Network (CDN)	Azure CDN	Amazon CloudFront	Cloud CDN
SQL Database Options	Azure SQL Database	Amazon RDS	Cloud SQL
NoSQL Database Options	Azure DocumentDB	AWS DynamoDB	Cloud Datastore
Virtual Network	Azure Virtual Network	Amazon VPC	Cloud Virtual Network
Private Connectivity	Azure Express Route	AWS Direct Connect	Cloud Interconnect
DNS Services	Azure Traffic Manager	Amazon Route 53	Cloud DNS
Log Monitoring	Azure Operational Insights	Amazon CloudTrail	Cloud Logging
Performance Monitoring	Azure Application Insights	Amazon CloudWatch	Stackdriver Monitoring
Administration and Security	Azure Active Directory	AWS Identity and Access Management (IAM)	Cloud Identity and Access Management (IAM)
Compliance	Azure Trust Center	AWS CloudHSM	Google Cloud Platform Security
Analytics	Azure Stream Analytics	Amazon Kinesis	Cloud Dataflow
Automation	Azure Automation	AWS Opsworks	Compute Engine Management
Management Services & Options	Azure Resource Manager	Amazon Cloudformation	Cloud Deployment Manager
Notifications	Azure Notifcation Hub	Amazon Simple Notification Service (SNS)	None
Load Balancing	Load Balancing for Azure	Elastic Load Balancing	Cloud Load Balancing

Most commonly used AWS Services to build data pipelines

Grow Data Skills

- S3
- Lambda
- IAM
- EVENT BRIDGE
- EC2
- SNS
- SQS
- STEP FUNCTIONS
- GLUE
- KINESIS
- RDS
- ATHENA
- REDSHIFT
- DynamoDB

AWS S3

Grow Data Skills

Amazon S3 (Simple Storage Service) is one of the foundational services in the AWS suite and is widely used by businesses and individuals to store and retrieve any amount of data, at any time, from anywhere.

Overview of AWS S3:

- **Object Storage:** S3 is an object storage service, meaning it is designed to store unstructured data (like photos, videos, backups, etc.) as objects within resources called "buckets".
- **Durability and Availability**: AWS S3 is designed for 99.999999999% (11 9's) durability over a given year. This ensures that your data remains safe and intact.
- **Scalability**: There's no limit to the amount of data you can store in S3, and it's designed to handle high request rates and traffic.
- **Data Organization**: Data in S3 is organized into buckets (similar to directories) and objects (files).
- **Versioning**: S3 supports versioning, allowing you to retain, retrieve, and restore every version of every object in your bucket.
- **Security**: Offers features like bucket policies, ACLs (Access Control Lists), and server-side encryption (SSE) for data. Integrated with AWS Identity and Access Management (IAM) for access control.
- **Event Configuration**: You can set up event notifications to trigger workflows, alerts, or other automated processes based on changes to your data.

AWS S3 Pricing Factors

Grow Data Skills

AWS S3 pricing is based on several factors:

- **Storage**: You're billed per GB per month based on the amount of data stored.
- **Requests**: Costs associated with the number and type of requests made (GET, PUT, COPY, etc.).
- **Data Transfer**: While transferring data into S3 is typically free, transferring data out of S3 to the internet or other AWS regions incurs charges.
- **Additional Features**: Features like versioning, monitoring with CloudWatch, data transfer acceleration, and others might have associated costs.
- **Storage Management**: Using features like S3 Inventory, S3 Analytics, and S3 Object Tagging will also influence the total cost.

S3 command with AWS CLI

Grow Data Skills

The AWS Command Line Interface (CLI) is a powerful tool that allows users to interact with AWS services, including S3, directly from the command line. Here's a list of some commonly used AWS S3 CLI commands:

- **Configuration:**
  - **aws configure**: Setup the CLI with your AWS credentials, default region, and desired output format.

- **Bucket Operations:**
  - **aws s3 ls:** List all buckets.
  - **aws s3 mb s3://my-bucket-name:** Create a new bucket.
  - **aws s3 rb s3://my-bucket-name:** Delete a bucket.

- **File and Folder Operations:**
  - **aws s3 ls s3://my-bucket-name:** List contents of a bucket.
  - **aws s3 cp localfile.txt s3://my-bucket-name/:** Copy a local file to a bucket.
  - **aws s3 cp s3://my-bucket-name/file.txt localfile.txt:** Copy a file from a bucket to the local system.
  - **aws s3 mv localfile.txt s3://my-bucket-name/:** Move a local file to a bucket (removes the local file after copying).
  - **aws s3 rm s3://my-bucket-name/file.txt:** Delete a file from a bucket.

AWS Lambda

Grow Data Skills

AWS Lambda is a serverless computing service that lets you run code without provisioning or managing servers. It automatically scales your application by running code in response to events, such as changes to data in Amazon S3 buckets or updates in an Amazon DynamoDB table.

Here are the primary features and properties of AWS Lambda:

- **Event-Driven**: AWS Lambda is designed to use events like changes to data in an S3 bucket or an update to a DynamoDB table to trigger the execution of code.
- **Scaling**: Lambda functions can scale automatically by running code in response to each trigger. Your trigger can be an uploaded image, a new log file, a new row in a database, etc.
- **Languages Supported**: As of my last update, AWS Lambda supports multiple programming languages. These include Node.js, Python, Ruby, Java, Go, .NET Core, and custom runtimes that you can provide.
- **Stateless**: By default, AWS Lambda is stateless, meaning each function execution is independent. If you need to maintain state, you would use an external service, like Amazon RDS or DynamoDB.
- **Short-lived**: Lambda functions are designed to be short-lived. Initially, there was a 5-minute max execution time, which later was extended to 15 minutes.
- **Resource Specification**: You can specify the amount of memory allocated to your Lambda function. AWS Lambda allocates CPU power linearly in proportion to the amount of memory configured.

AWS Lambda

- **Built-in Fault Tolerance**: AWS Lambda maintains compute capacity and infrastructure reliability, including monitoring, logging via Amazon CloudWatch, and automatic retries.
- **Deployment**: Code can be deployed as a Lambda function via a ZIP or JAR file. AWS also provides a blueprints feature to start off with sample code for common use cases.
- **Integrated with AWS Services**: It's integrated with many AWS services making it a flexible tool. For instance, you can trigger a Lambda function from changes in S3, updates in DynamoDB, endpoint requests in API Gateway, etc.
- **Layers**: Lambda Layers are a distribution mechanism for libraries, custom runtimes, and other function dependencies. Layers promote code sharing and separation of responsibilities.
- **Billing**: With AWS Lambda, you're billed for the compute time your code is running. You aren’t charged when your code isn’t running.
- **Event Source Mapping:** If a Lambda function is triggered by an event source, AWS Lambda takes care of the reading, retries, and deletion of the event, ensuring that each event is processed in order.
- **Concurrent Executions**: AWS Lambda scales functions in parallel. While it manages and scales these automatically, there is a default safety throttle for the number of concurrent executions across all functions in a given region.

AWS Lambda

Grow Data Skills

AWS SNS

Grow Data Skills

Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service provided by AWS. It is designed for distributing notifications to a wide range of recipients. With SNS, you can send messages to individual recipients or to large numbers of recipients.

Key Features and Properties of AWS SNS:

- **Pub/Sub Messaging**: SNS follows the publish/subscribe (pub/sub) messaging paradigm, allowing users to create "topics" and then have subscribers that receive messages or notifications on those topics.

- **Multiple Protocols**: SNS supports multiple protocols, meaning you can deliver messages to:
  - HTTP/HTTPS endpoints
  - Email/Email-JSON
  - Short Message Service (SMS)
  - Application (for sending messages to other AWS services or to applications)
  - AWS Lambda
  - Simple Queue Service (SQS)
  - Application Endpoints (for mobile devices)

- **Flexibility**: You can send a message to an SNS topic, and then that single message can be delivered to many recipients across various supported protocols.

- **Durability**: SNS messages are stored redundantly across multiple servers and data centers, providing high availability and durability.

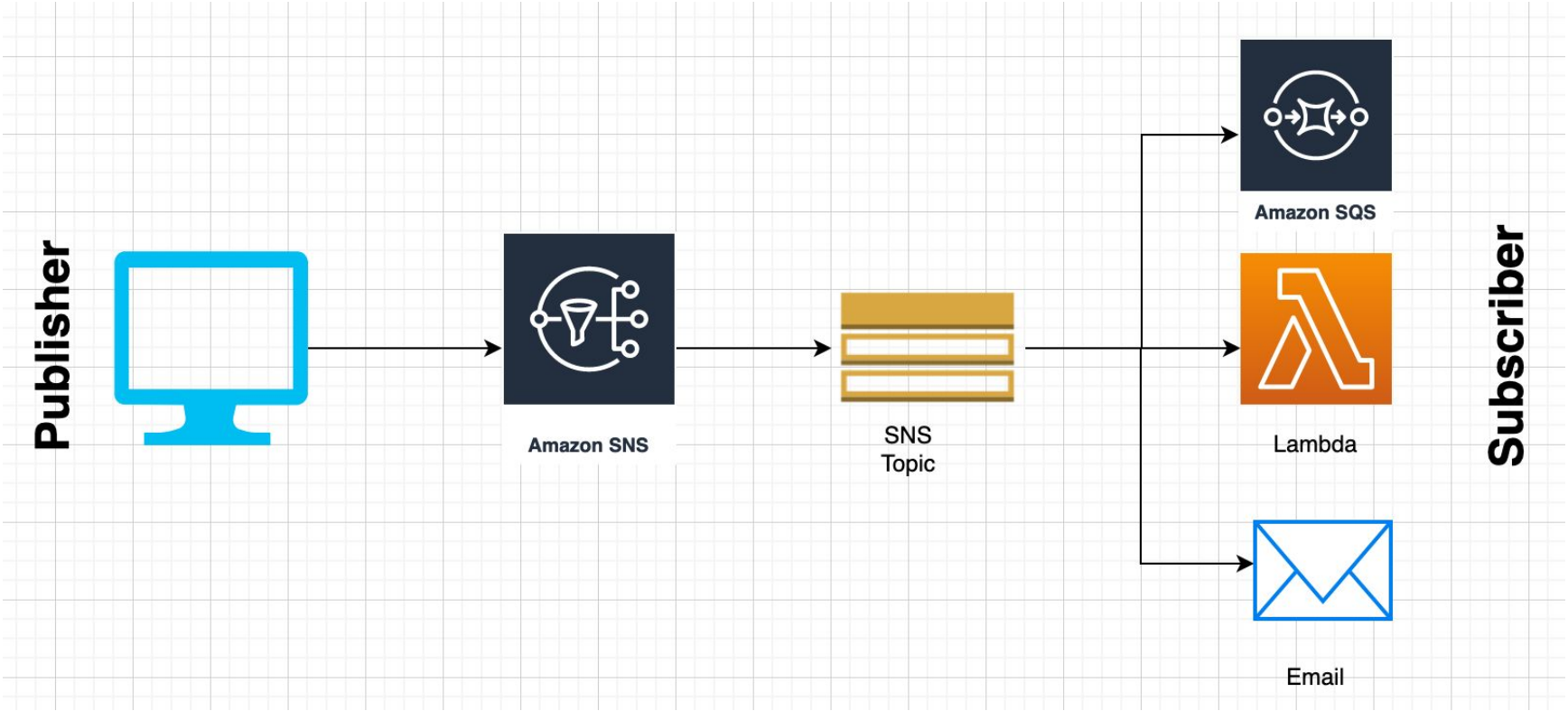
AWS SNS

Grow Data Skills

- **Content Filtering**: With SNS, you can filter the messages delivered to each subscription, ensuring subscribers only receive the messages of interest to them.
- **Access Control**: Integration with AWS Identity and Access Management (IAM) allows granular access control to the SNS topics. You can specify who can publish or subscribe to a topic.
- **Large Message Size**: For messages that exceed the normal size limit (256 KB), SNS can store the large message in an Amazon S3 bucket and send a pointer to the message in the notification.
- **Monitoring**: Integrated with Amazon CloudWatch, allowing users to monitor metrics related to the SNS service.
- **Encryption**: Supports encryption in transit (using HTTPS endpoints) and at rest (using AWS Key Management Service).
- **Cost**: Users pay for what they use. This includes the number of requests, number of messages delivered, and data transfer. There's no upfront commitment required.

AWS SNS

Grow Data Skills



AWS SQS

Grow Data Skills

Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that enables the decoupling of microservices, distributed systems, and serverless applications. It is designed to send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.

Key Features and Properties of AWS SQS:

- **Fully Managed**: No need to manage and operate message-oriented middleware systems or any other infrastructure.
- **Durability**: SQS stores messages redundantly across multiple servers and data centers to ensure that a message is delivered at least once.
- **Two Queue Types**:
  - **Standard Queue**: Offers maximum throughput, best-effort ordering, and at-least-once delivery.
  - FIFO (First-In-First-Out) Queue: Ensures messages are processed only once, in the exact order they are sent.
- **Message Attributes**: Messages can contain metadata (as key-value pairs) so that the receiver can handle the message appropriately.
- **Long Polling:** Reduces unnecessary network traffic with empty responses by waiting until a message is available in the queue before sending a response.

AWS SQS

- **Batch Operations**: Supports sending, receiving, or deleting messages in batches, helping to improve the efficiency of both your producer and consumer applications.
- **Dead Letter Queues**: If a message cannot be processed successfully, it's moved to a dead letter queue. This is useful for debugging and ensuring that problematic messages don't get stuck and aren't retried indefinitely.
- **Visibility Timeout**: After a message is retrieved by a consumer, it remains hidden from other consumers for a specified period. If the message isn't processed within that time, it becomes visible again.
- **Message Lifecycle**: You can set a retention period for messages, up to a maximum of 14 days.
- **Integration with other AWS Services**: SQS can be integrated with services like AWS Lambda, Amazon S3, and Amazon Redshift, among others.
- **Scalability**: SQS can handle high volumes of messages without any throughput limitations.
- **Server-Side Encryption (SSE)**: Uses AWS Key Management Service (AWS KMS) to encrypt SQS messages.

AWS SQS and Kafka are same?

Grow Data Skills

Amazon SQS and Apache Kafka address different messaging patterns and use cases, so it's not accurate to say SQS is a direct replacement for Kafka in AWS. However, they do overlap in some functionalities.

Here's a comparison to clarify:

- **Messaging Patterns:**
  - SQS: Primarily a message queue service designed for decoupling point-to-point communication between producers and consumers.
  - Kafka: A distributed streaming platform that can handle high-velocity data streams and allows for publish-subscribe and record storage.

- **Throughput**:
  - SQS: Suitable for a wide range of workloads, including those that require high throughput.
  - Kafka: Built for extremely high throughput and low latency, making it suitable for real-time analytics and monitoring.

- **Consumers:**
  - SQS: Each message is processed by a single consumer.
  - Kafka: Multiple consumers can read the same message from a topic without affecting other consumers.

- **Retention**:
  - SQS: Messages can be retained for a limited period (up to 14 days).
  - Kafka: Messages can be retained indefinitely or for a configured duration.

AWS SQS and Kafka are same?

Grow Data Skills

- **Ordering:**
  - SQS: Standard queues offer at-most-once delivery and best-effort ordering. FIFO queues offer exactly-once processing and guaranteed ordering.
  - Kafka: Provides strict ordering within a partition of a topic.

- **Scalability:**
  - SQS: Managed by AWS and scales automatically.
  - Kafka: Requires manual cluster scaling and configuration.

- **Ecosystem**:
  - SQS: Integrated tightly within the AWS ecosystem.
  - Kafka: Has a vast ecosystem with Kafka Streams, Kafka Connect, and more. It's more than just a messaging system; it's a whole streaming platform.

If you're looking for a managed Kafka-like service in AWS, consider **Amazon Managed Streaming for Apache Kafka (Amazon MSK)**. It's a fully managed service that makes it easy to build and run applications that use Apache Kafka to process streaming data. With Amazon MSK, you get the combined capabilities and benefits of Apache Kafka along with the scalability and reliability of AWS.

AWS SQS

Grow Data Skills