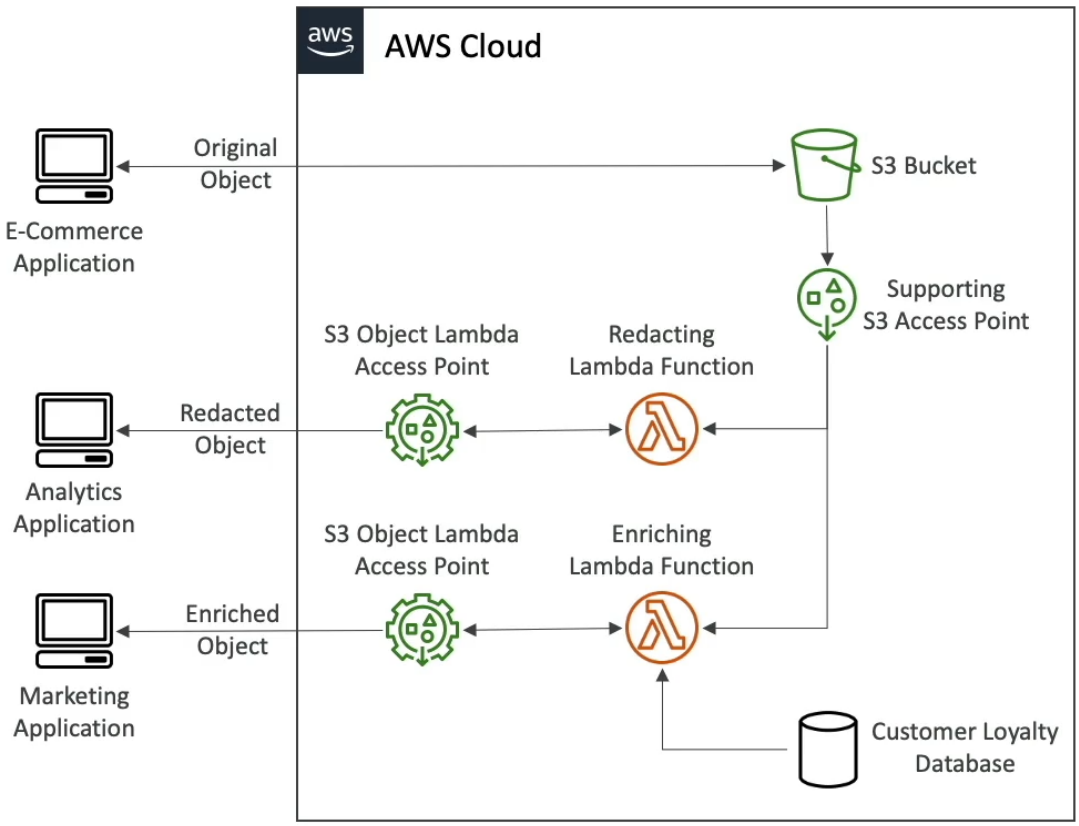


Use cases:

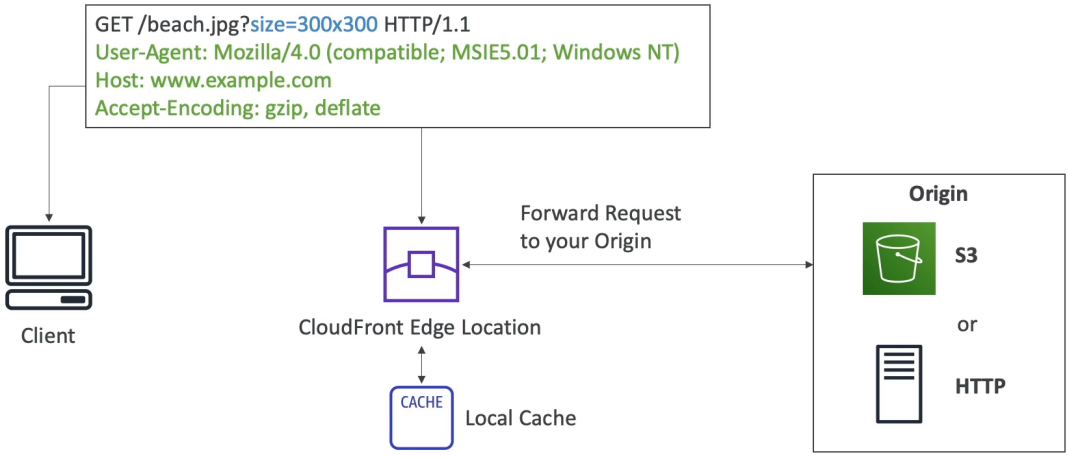
- Redacting PII information from analytics or non-productiuon environments.
- Converting across data formats, such as converting XML to JSON.
- Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.

CloudFront, Global Accelerator & Wavelength

▼ ☑ **AWS CloudFront.**

AWS CloudFront is a Content Delivery Network (CDN) service. It delivers our content to end-users with low latency and high transfer speeds.. It improves read performance, content is cached at the edge location. When a user requests content, CloudFront serves it from the nearest edge location.

CloudFront integrates with AWS Shield for DDoS protection, AWS Web Application Firewall (WAF) to protect against common web exploits, and SSL/TLS encryption to secure data transfer.



CloudFront Access Control

CloudFront signed URLs and signed cookies allow us to control who can access our content.

Signed URLs use cases:

- RTMP distribution (signed cookies aren't supported for RTMP distributions).
- Restrict access to individual files (e.g. an installation download for your application).
- Users are using a client that doesn't support cookies.

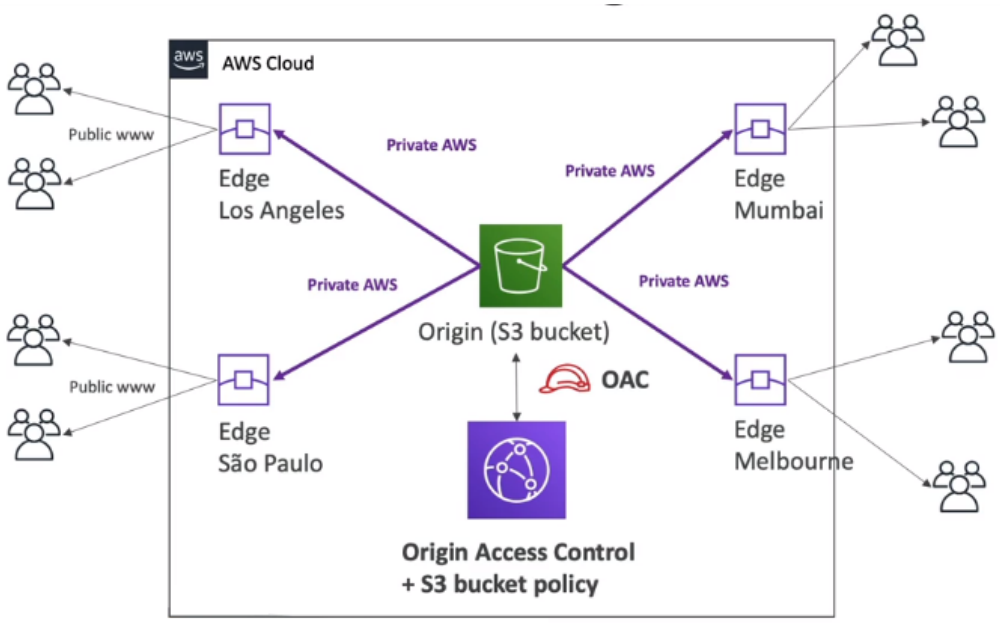
Signed Cookies use cases:

- Provide access to multiple restricted files (e.g. all of the files for a video in HLS format or all of the files in the subscribers' area of a website).
- We don't want to change our current URLs.

CloudFront Origins

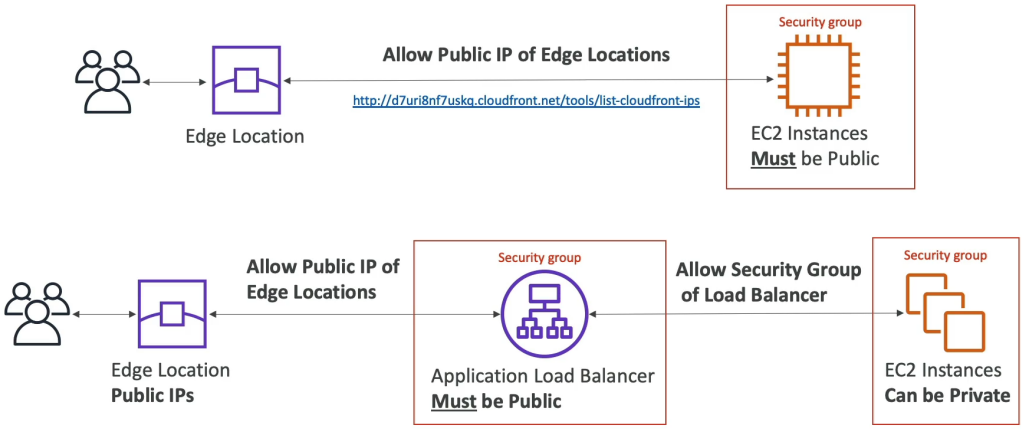
- S3 bucket

Used for distributing files and caching them at the edge location. Enhanced security with CloudFront **Origin Access Control (OAC)**. It can be used as an ingress (to upload files to S3).



- Custom Origin (HTTP)

  It can be **ALB, EC2 instance or S3 website** (must enable the bucket as a static S3 website).



CloudFront vs S3 Cross Region Replication

- CloudFront:
  - Global Edge network
  - Files are cached for a TTL (maybe a day)
  - Great for static content that must be available everywhere

- S3 Cross Region Replication:
  - Must be setup for each region you want replication to happen
  - Files are updated in near real-time
  - Read only
  - Great for dynamic content that needs to be available at low-latency in few regions

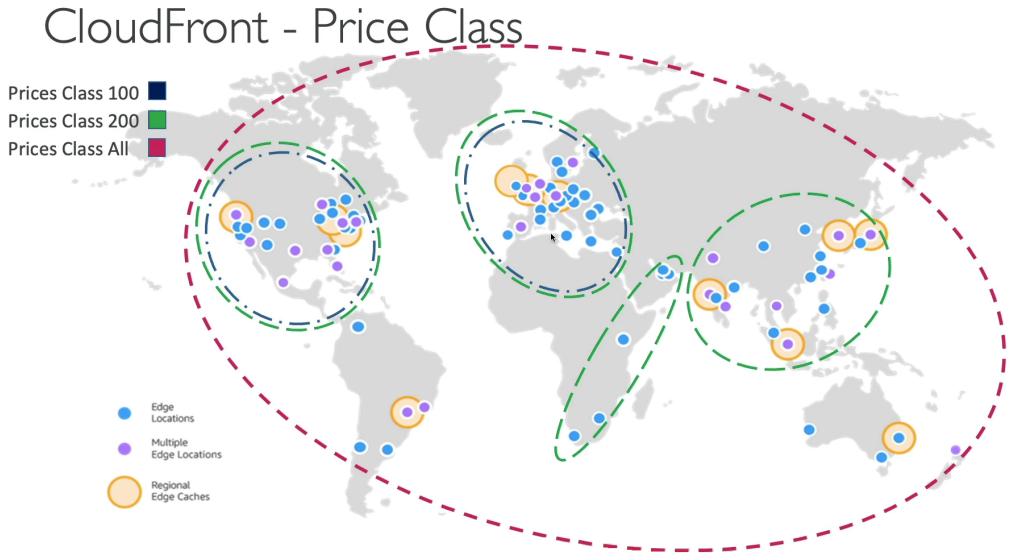
Origin Access Identity (OAI) is a special CloudFront user that we can create to give CloudFront permission to access our content stored in an S3 bucket.

▼ ☑ **CloudFront Geo Restriction.**

We can restrict who can access our distribution based on geographic location. The country is determined using a third party Geo-IP database.

- **Allowlist** - allow our users to access our content only if they are in one of the countries on a list of approved countries.
- **Blocklist** - prevent our users from accessing our content if they are in one of the countries on a list of banned countries.

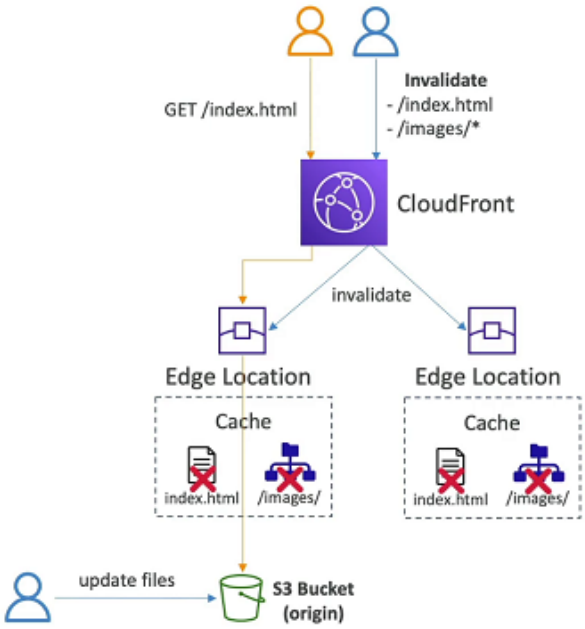
▼ ☑ **CloudFront Price Classes.**



▼ ☑ **CloudFront Cache Invalidation.**

The Cache-Control and Expires headers control how long objects stay in the cache. The **Cache-Control max-age** directive lets us specify how long (in seconds) we want an object to remain in the cache before CloudFront gets the object again from the origin server. The minimum expiration time CloudFront supports is 0 seconds for web distributions and 3600 seconds for RTMP distributions.

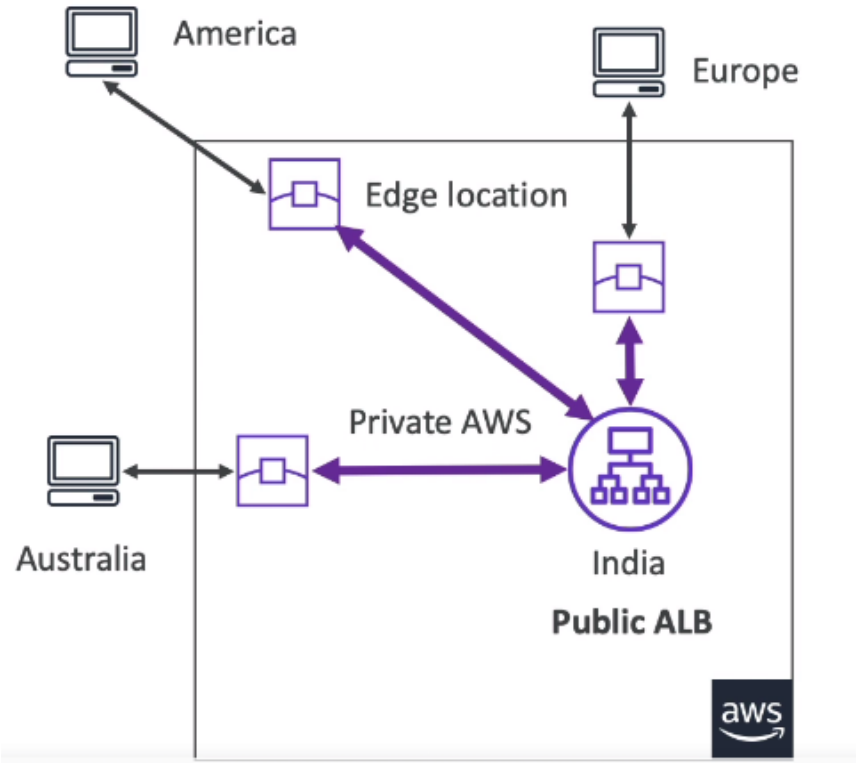
In case we update the back-end origin, CloudFront doesnt know about it and will only get the refreshed content after the TTL has expired. We can force an entire or partial cache refresh by performing a CloudFront Invalidation. We can invalidate all files or special path.



▼ ☑ **AWS Global Accelerator.**

AWS Global Accelerator - improves global application availability and performance using the AWS global network. It leverages the AWS internal network to optimize the route to our application (60% improvement).

It provides two static IP addresses that serve as a fixed entry point to our application. These IP addresses are anycast from AWS edge locations, ensuring user traffic is directed to the closest AWS region.



It works with Elastic IP, EC2 instances, ALB, NLB. We can use AWS Shield for DDoS protection.

Global Accelerator performs a health check of our applications. It helps make our application global (failover less than 1 minute for unhealthy) and it's great for disaster recovery.

AWS Global Accelerator vs CloudFront

- They both use the AWS global network and its edge locations around the world
- Both services integrate with AWS Shield for DDoS protection.
- CloudFront – Content Delivery Network
  - Improves performance for your cacheable content (such as images and videos)
  - Content is served at the edge

- Global Accelerator
  - No caching, proxying packets at the edge to applications running in one or more AWS Regions.
  - Improves performance for a wide range of applications over TCP or UDP
  - Good for HTTP use cases that require static IP addresses
  - Good for HTTP use cases that required deterministic, fast regional failover

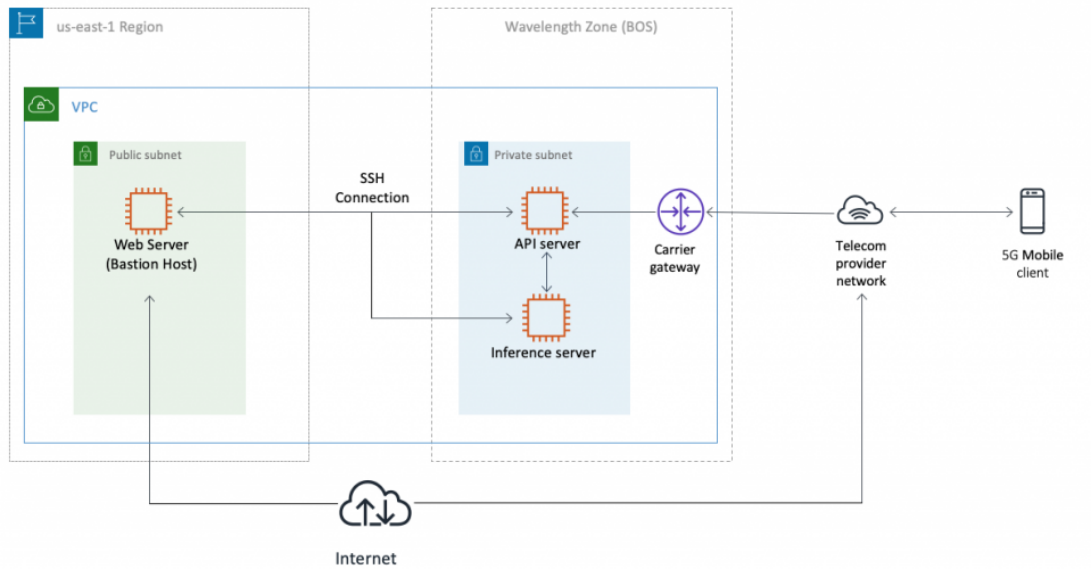
▼ ☑ AWS Wavelength.

AWS Wavelength is a service that brings AWS compute and storage services to the edge of the 5G network. It enables developers to build ultra-low-latency applications by deploying them closer to end-users, specifically within the infrastructure of telecom providers.

Wavelength Zones are specific infrastructure deployments that host AWS compute and storage services within the 5G networks. These zones are deployed in partnership with telecom providers like Verizon, Vodafone, KDDI, and SK Telecom, among others. We can extend our VPC into Wavelength Zones.

Carrier Gateway - networking component used specifically in Wavelength Zones to enable communication between our VPC and devices connected to 5G networks. It provides a route for traffic between your VPC and the telecom provider's 5G network, allowing your Wavelength-hosted applications to communicate with devices using carrier-assigned IP addresses.

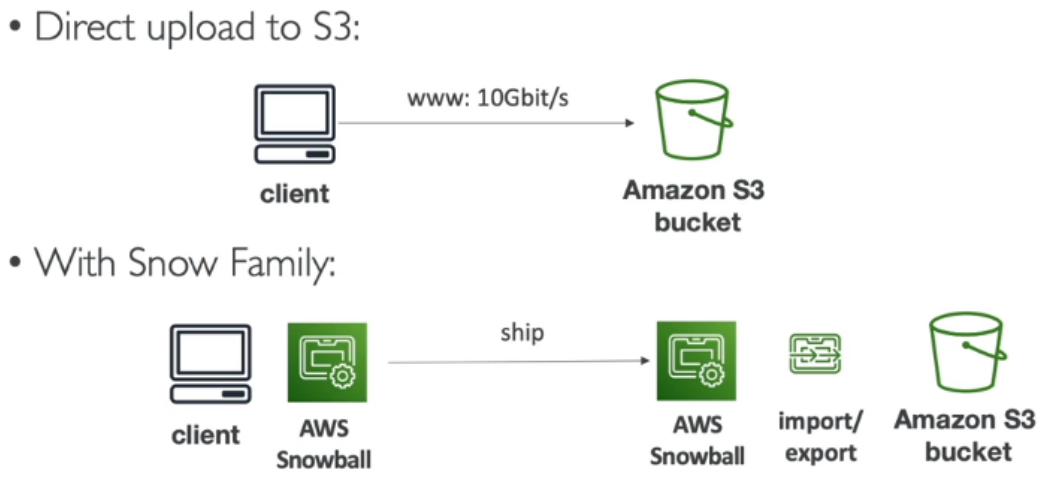
Communication between applications and 5G devices can remain private within the telecom provider's network without ever needing to cross the public internet.



AWS Storage Extras

▼ ☑ **AWS Snow Family.**

AWS Snow Family are highly-secure, portable devices which collect and process data at the edge and migrate data into and out of AWS. If it takes more than a week to transfer over the network, we should use Snowball devices.



- **Snowball Edge**

  This a physical data transport solution. We can move TBs or PBs of data in or out of AWS. It's alternative to moving data over the network and paying network fees. Instead we pay per data transfer job. Typical use cases are large data cloud migrations and disaster recovery.

  Snowball Edge proivides **block storage** and Amazon **S3** compatibile object storage.

  - **Snowball Edge Storage Optimized**

    80TB of HDD or 210TB NVMe capacity for block volume and S3 compatible object storage.

  - **Snowball Edge Compute Optimized**

    42TB of HDD or 28TB NVMe capacity for block volume and S3 compatible object storage.

- **Snowcone**

  It's a small, portable, rugged & secure device. Withstands hars enviroments. Typically used for edge computing, storage and data transfer. When Snowball does not fit we should use Snowcone (we must provide our own battery and cables). It can be sent back to AWS offline or connect it to internet and use AWS DataSync to send data.

  - **Snowcone** - 8TB of HDD Storage

    - **Snowcone SSD** - 14TB od SSD Storage
- **Snowmobile**

  It's a truck which can transfer exabytes of data. Each Snowmobile has 100PB capacity. If we transfer more thatn 10PB it is better than Snowball.

Snowcone | Snowball Edge | Snowmobile

	Snowcone & Snowcone SSD	Snowball Edge Storage Optimized	Snowmobile
Storage Capacity	8 TB HDD 14 TB SSD	80 TB - 210 TB	< 100 PB
Migration Size	Up to 24 TB, online and offline	Up to petabytes, offline	Up to exabytes, offline
DataSync agent	Pre-installed		

Snowbal Edge & Snowcone can be also used in edge computing (processing data while it's being created on an edge location). Use cases of edge computing: preprocessing data, ML, transcoding media streams ...

- **Snowcone & Snowcone SSD** ⇒ 2CPUs, 4GB of RAM, can use USB-C.
- **Snowball Edge - Compute Optimized** ⇒ 104CPUs, 416GB of RAM, optional GPU, 28TB NVMe or 42TB HDD.
- **Snowball Edge - Storage Optimized** ⇒ 40CPUs, 80GB of RAM, 80TB storage OR 104CPUs, 416GB of RAM, 210TB NVMe storage

All can run EC2 Instances & AWS Lambda functions (using AWS IoT Greengrass). There are long-term deployment options - 1 and 3 years discounted pricing. Snowcone is better for IoT.

▼ **Solution Architecture - Snowball into Glacier**

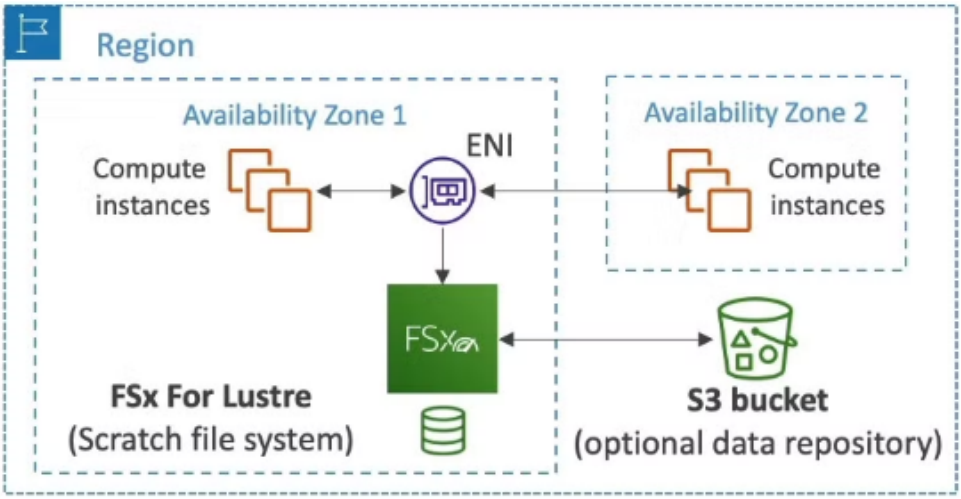
Snowball cannot import to Glacier directly. We must use S3 first in combination with an S3 lifecycle policy.

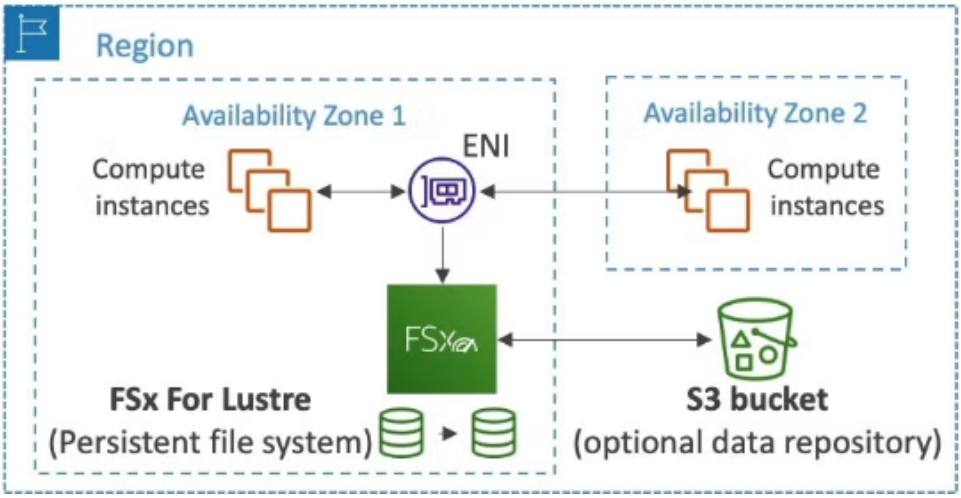
▼ ☑ **Amazon FSx.**

Amazon FSx is a fully managed service that lets us launch third party high-performance file systems on AWS.

FSx System Deployment Options

- **Scratch File System** - temporary storage. Data is not replicated (doesn't persist if file server fails). It has high burst and it is used for short-term processing.
- **Persistent File System** - long-term storage. Data is replicated within same AZ. It replaces failed files within minutes. Used for long-term processing and sensitive data.



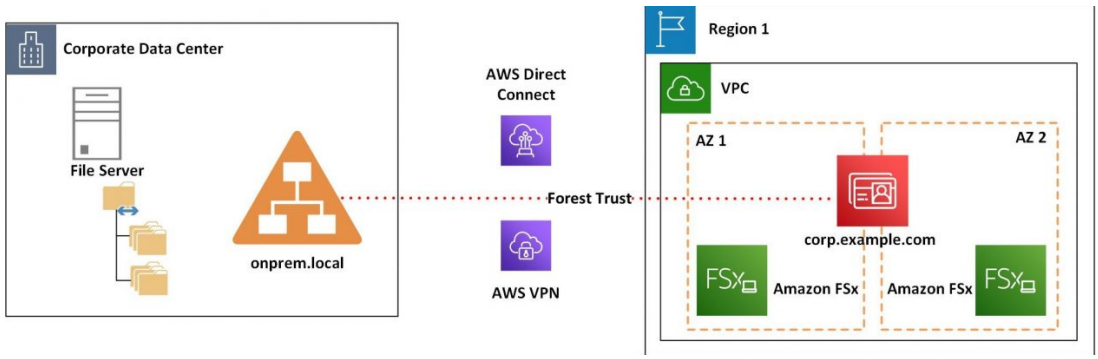


▼ ☑ **FSx - Windows File Server.**

FSx for Windows File Server - supports SMB and NTFS protocols. It can be mounted on Linux EC2 instances. We can integrate it with Microsoft AD, ACLs and user quotas. It supports Microsoft's Distributed File System (DFS) Namespaces.

We can choose SSD or HDD for storage options. FSx for Windows can be accessed from our on-premises infrastructure (VPN or DX). Can be configured to be Multi-AZ and all data is backed-up daily to S3.

FSx for Windows File Server **can be joined to an existing on-premises Active Directory**. This allows the file system to continue using the same AD groups and policies to restrict access to the shares, folders, and files, just as it did on-premises.

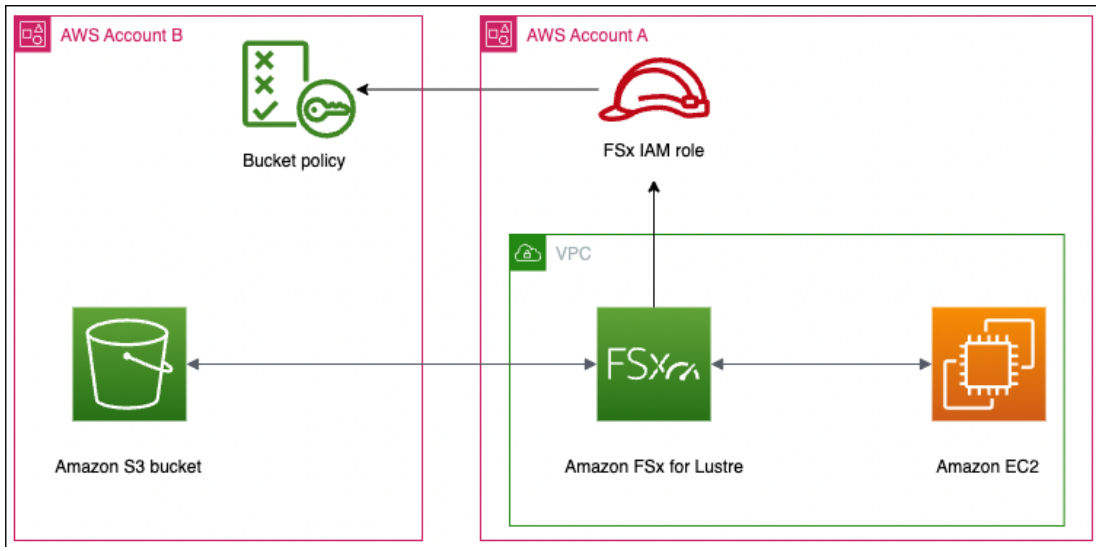


▼ ☑ **FSx - Lustre.**

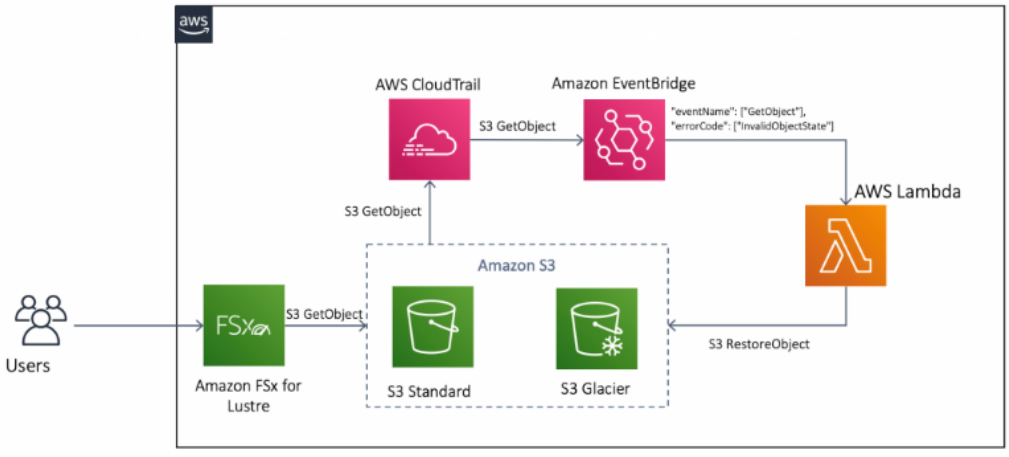
FSx for Lustre - type of **parallel** distributed file system, for large-scale computing. It is commonly used for HPC and uses ML. We can chose between SSD and HDD storage options. Can be used from on-premises servers (VPN or Direct Connect).

FSx for Lustre uses a striped architecture, where data is distributed across multiple disks to optimize performance.

We can read S3 as a file system through FSx and write the output of the computations back to S3 (through FSx).



▼ **Solution Architecture - Automate File Retrieval from S3 Glacier.**

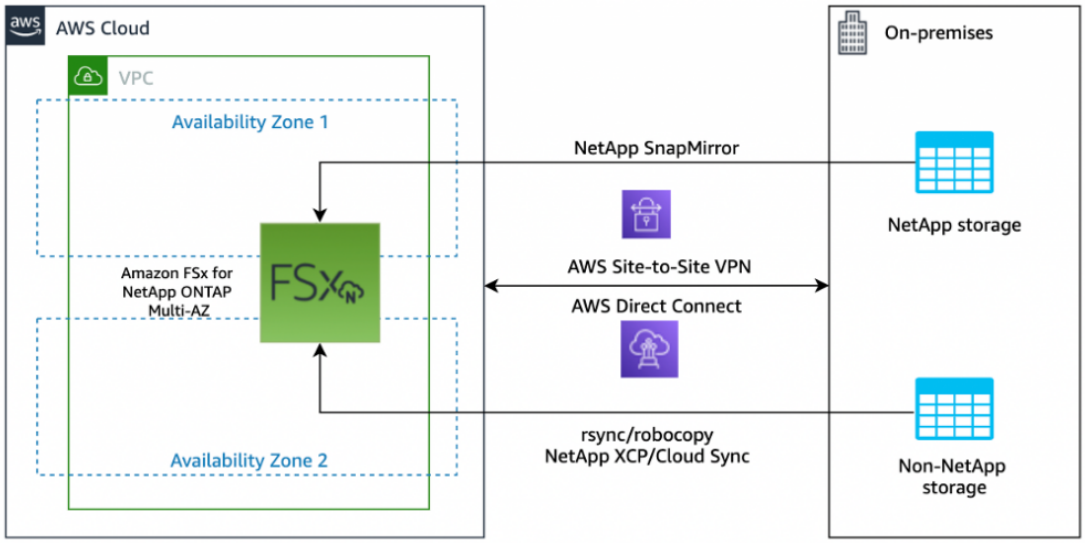


▼ ☑ **FSx - NetApp ONTAP & OpenZFS.**

FSx for NetApp ONTAP - managed NetApp ONTAP on AWS. It moves workloads running on ONTAP or NAS to AWS. It is compatible with NFS, SMB, iSCSI protocol.

Its storage shrinks or grows automatically and we can do snapshots, replication, data de-duplication and compression. We can do point-in-time instantaneous cloning (helpful for testing new workloads).

ONTAP SnapMirror - a replication technology in ONTAP that allows us to asynchronously replicate data between two storage systems.



FSx for OpenZFS - managed OpenZFS file system on AWS. It is compatible with NFS and moves workloads running on ZFS to AWS. We can do point-in-time instantaneous cloning. Also can do snapshots and compression.

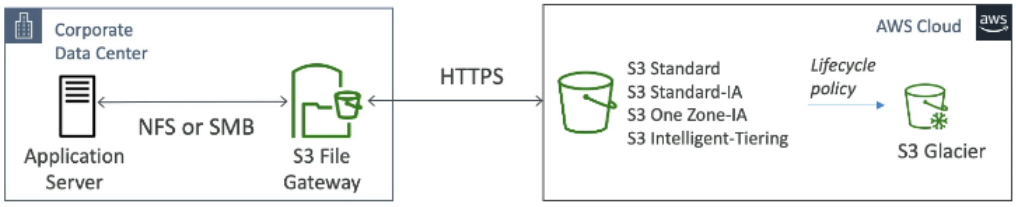
▼ ☑ **AWS Storage Gateway.**

S3 is a proprietary storage technology and only way to expose S3 data on-premise is using AWS Storage Gateway.

AWS Storage Gateway is a bridge between on-premise data and cloud data in S3. Our hybrid storage service will allow on-premises to seamlessly use the AWS Cloud (disaster recovery, backup & restore, tiered storage).

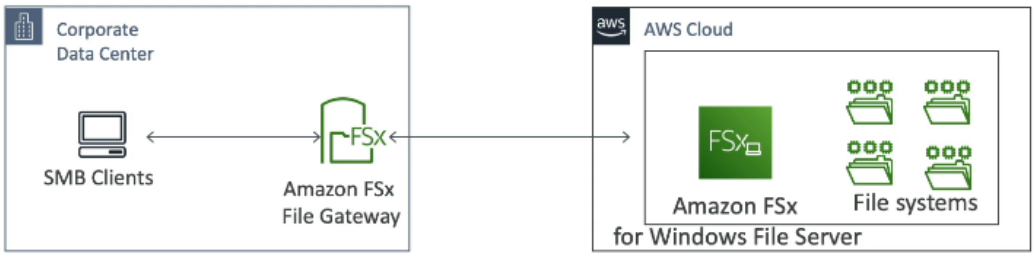
Types of Storage Gateway:

- **S3 File Gateway** - offers file-based access to objects in Amazon S3. It allows us to store and retrieve objects in Amazon S3 using file protocols, such as **NFS and SMB**. Most recently used data is cached in the file gateway. It supports all S3 classes except S3 Glacier. Transition to S3 Glacier must be done using a Lifecycle Policy.



To access our buckets we must create IAM role for each File Gateway. Also if we use SMB we can use its integration with Active Directory for user authentication.

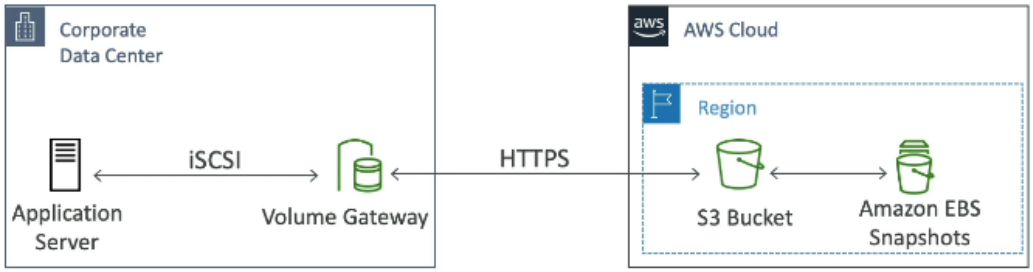
- **FSx File Gateway** - offers native access to Amazon FSx for Windows File Server. It has local cache for frequently accessed data.



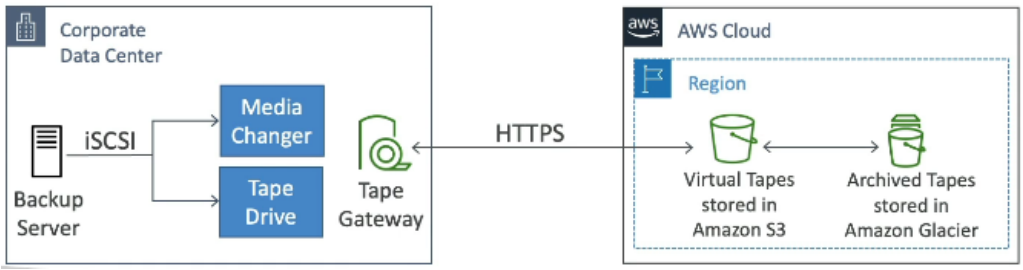
- **Volume Gateway** - provides block storage to on-premises applications using iSCSI protocol, with data asynchronously backed up to AWS.

  There are two modes for Volume Gateway:

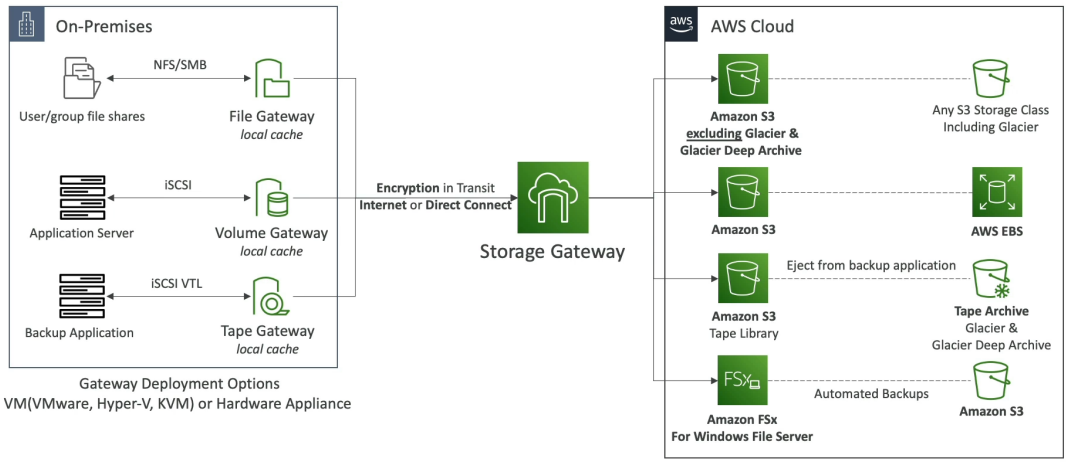
  - **Cached Volumes** - frequently accessed data is cached locally, while the primary data is stored in S3.
  - **Stored Volumes** - entire datasets are stored locally, with scheduled backups to S3.



- **Tape Gateway** - enables us to use Amazon S3 and Glacier as a scalable, cost-effective solution for archival and backup data stored on physical tapes. It presents a virtual tape library (VTL) interface, compatible with leading backup software.



Using Storage Gateway means we need on-premises virtualization. Otherwise, we can use a Storage Gateway Hardware Appliance. It has the required CPU, memory, network and SSD cache resources.



▼ ☑ **AWS Transfer Family.**

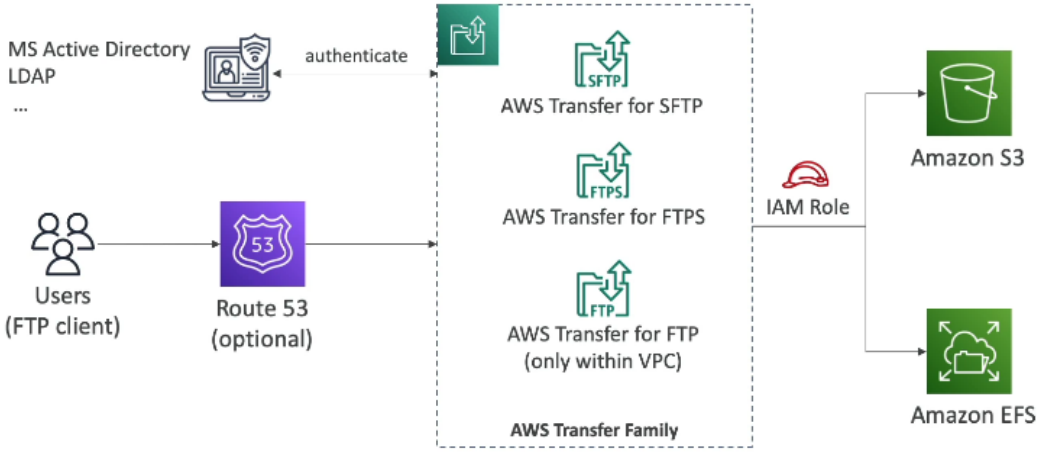
Transfer Family is a set of fully managed services for file transfers into and out of S3 or EFS using the FTP protocol. We pay per provisioned endpoint per hour and data transfers in GB. It is possible to store and manage users' credentials within the service.

- **AWS Transfer for FTP**
- **AWS Transfer for FTPS**
- **AWS Transfer for SFTP**

Can be integrated with existing authentication systems.

With Transfer Family's AS2 capabilities, we can securely exchange AS2 messages at scale while maintaining compliance and interoperability with trading partners. **Applicability Statement 2 (AS2)** is a business-to-business (B2B) messaging protocol used to exchange Electronic Data Interchange (EDI) documents.

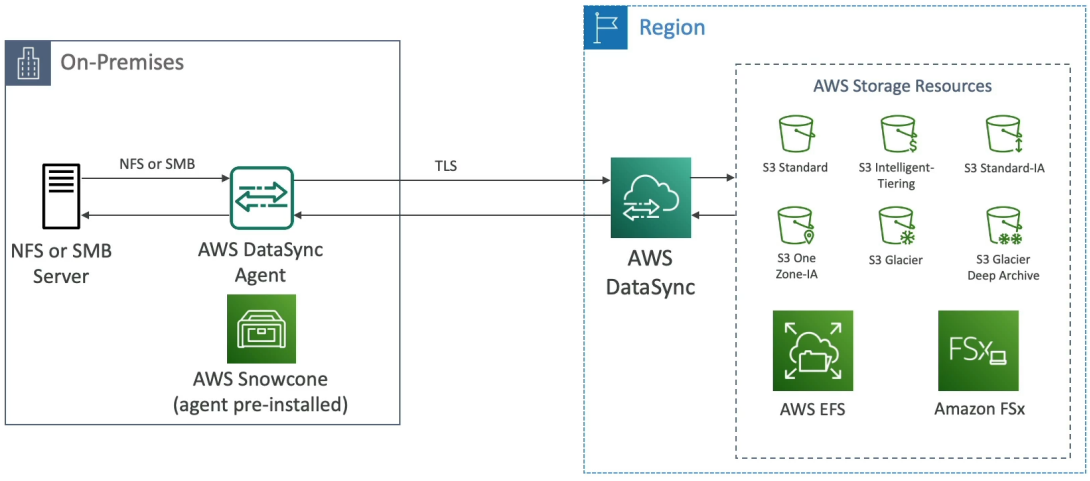
Use cases: sharing files, public datasets, CRM, ERP (Enterprise Resource Planing).



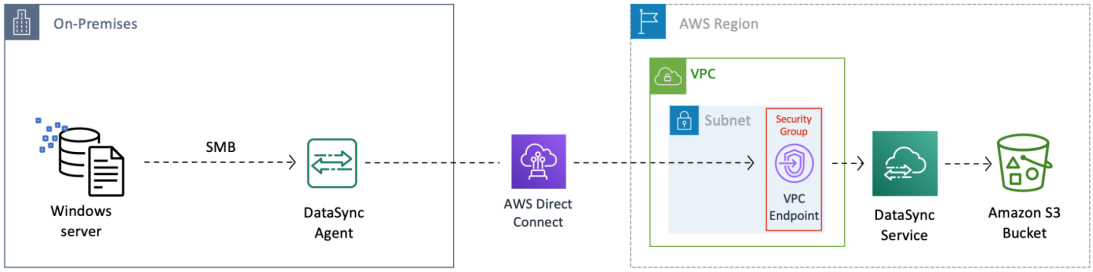
▼ ☑ **AWS DataSync.**

DataSync is used to automate and accelerate the replication of data and move large amount of data to and from on-premises to AWS (need agent) or AWS to AWS. It can synchronize to **S3, EFS** and **FSx**. Replication tasks can be scheduled hourly, daily or weekly.

File permissions and metadata are preserved (NFS POSIX, SMB...). One agent task can use 10 Gbps and we can setup a bandwidth limit.



The DataSync agent can route DataSync traffic from the on-premises storage system (source location) to the DX connection.



Decoupling Applications

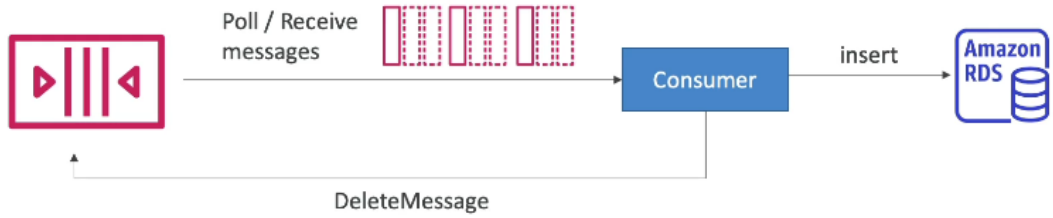
▼ ☑ **AWS SQS - Standard Queues.**

SQS is a fully managed message queuing service that enables us to decouple and scale microservices, distributed systems, and serverless applications.

There is no limit how many messages can be in the queue. Default retention of messages is 4 days, maximum of 14 days. Messages in the SQS queue will continue to exist even after the EC2 instance has processed it, until we delete that message. Consumers share the work to read messages and scale horizontally. It can have duplicate messages.

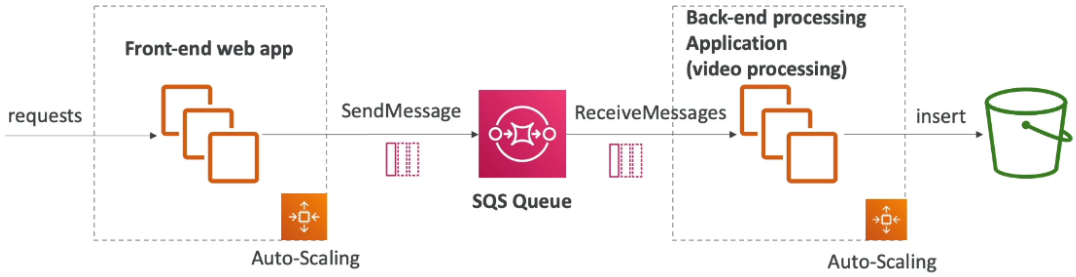
We can use SendMessage API by AWS SDK to produce messages. The message is **persisted** in SQS until a consumer deletes it.

Message consumers are running on EC2 instances, servers or AWS Lambda. They poll SQS for messages, process the messages and then delete them using DeleteMessage API.



Consumers receive and process messages in parallel and we can scale consumers horizontally to improve throughput of processing.

Decouple between Application Tiers



SQS Security

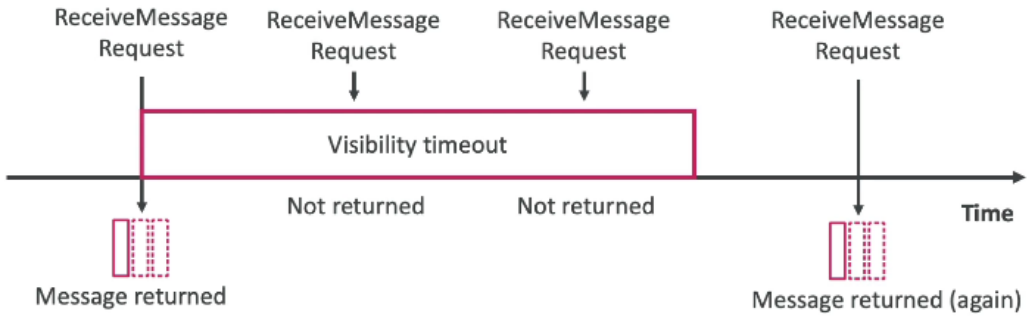
Encryption - has in-flight encryption using HTTPS, at-rest encryption using KMS and we can do client-side encryption.

Access Controls - we can configure IAM policies to regulate access to the SQS API.

SQS Access Policies - similar to S3 bucket policies. Useful for cross-account access to SQS queues. They allow other services to write to an SQS queue.

▼ ☑ **SQS Message Visibility Timeout.**

After a message is polled by a consumer, it becomes invisible to other consumers. By default, the message visibility timeout is 30s (max 12h). After the message visibility timeout is over, the message is visible in SQS.

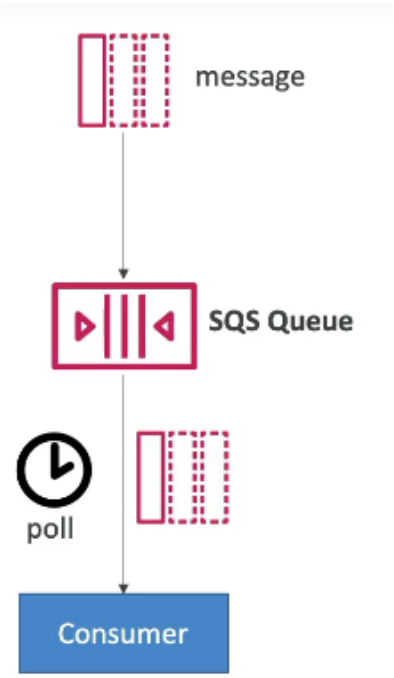


If a message is not processed within the visibility timeout, it will be processed twice. A consumer could call the ChangeMessageVisibility API to get more time.

If visibility timeout is high (hours), and consumer crashes, re-processing will take time and if visibility timeout is too low (seconds), we may get duplicates.

▼ ☑ **SQS Long Polling.**

Long Polling - when a consumer requests messages from the queue, it can optionally wait for messages to arrive if there are none in the queue.



Long Polling decreases the number of API calls made to SQS while increasing the efficiency and latency of our application. The **wait time can be between 1-20 seconds**.

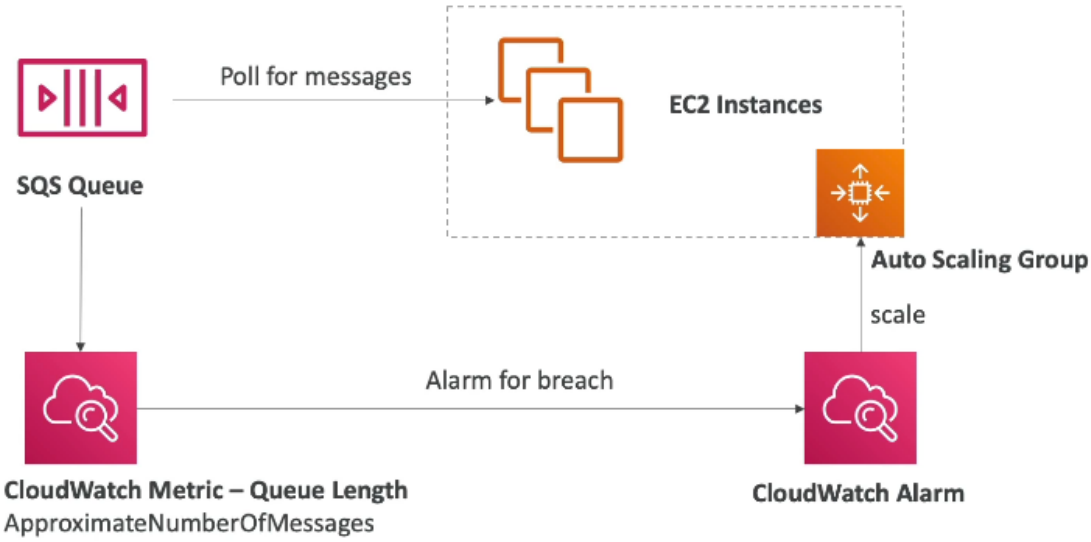
It can be enabled at the queue level or at the API level using **WaitTimeSeconds**.

▼ ☑ **AWS SQS - FIFO Queues.**

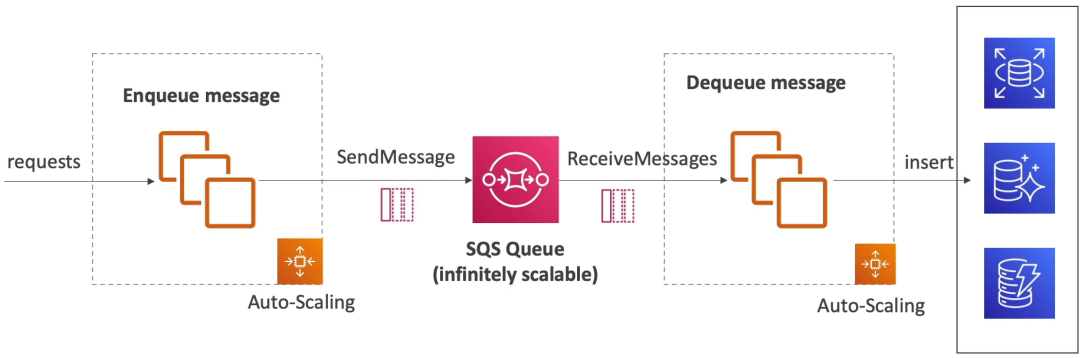
In Standard SQS messages are processed in random order by the consumer. If we want ordering of messages in the queue there is **SQS FIFO Queue** variant. There messages are processed in order by the consumer.

▼ ☑ **SQS + ASG Design Patterns.**

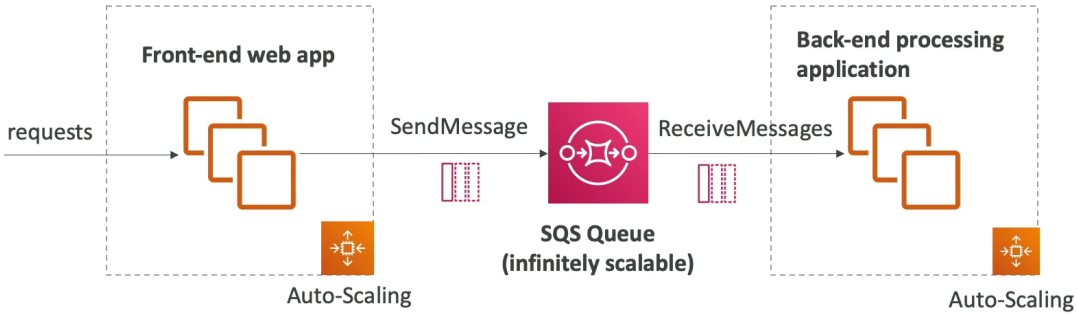
Scale Based on CloudWatch Alarm



Buffer to Database Writes

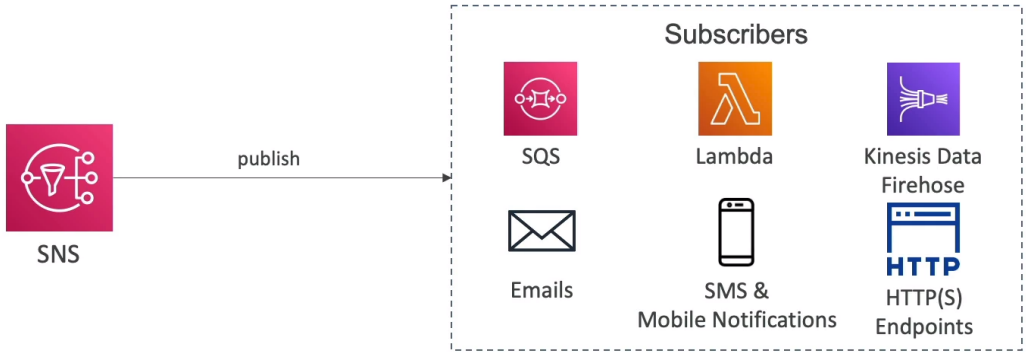


Decouple between Application Tiers



▼ ☑ **AWS SNS.**

SNS is a topic-based publisher/subscriber model. The event publishers only sends message to one SNS topic. Subscribers receive messages by subscribing to a topic. Subscribers can be various endpoints such as email, SMS, AWS Lambda functions. Each subscriber to the topic will get all the messages.



- **Topic Publish** - using SDK. First create topic, then create a subscription and publish to the topic.
- **Direct Publish** - for mobile apps SDK. We need to create a platform application and platform endpoint. Then publish to the platform endpoint.

SNS Message Filtering - assign a filter policy to the topic subscription, and the subscriber will only receive a message that they are interested in.

SNS has **FIFO Topic** (ordering of messages in the topic). It has similar features as SQS FIFO:

- **Ordering** by Message Gorup ID.

- **Deduplication** using a Deduplication ID or Content Based Deduplicaton.

Both SQS Standard and FIFO queues can be subscribers.

SNS Security

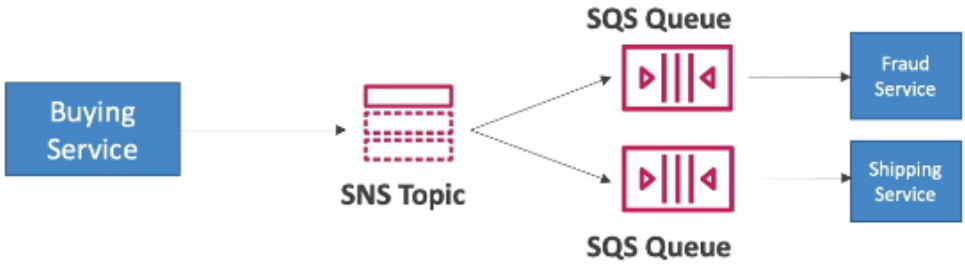
Encryption - has in-flight encryption using HTTPS, at-rest encryption using KMS and we can do client-side encryption.

Access Controls - we can configure IAM policies to regulate access to the SNS API.

SNS Access Policies - similar to S3 bucket policies. Useful for cross-account access to SNS topics. They allow other services to write to an SNS topic.

☑ SNS & SQS - Fan Out Pattern.

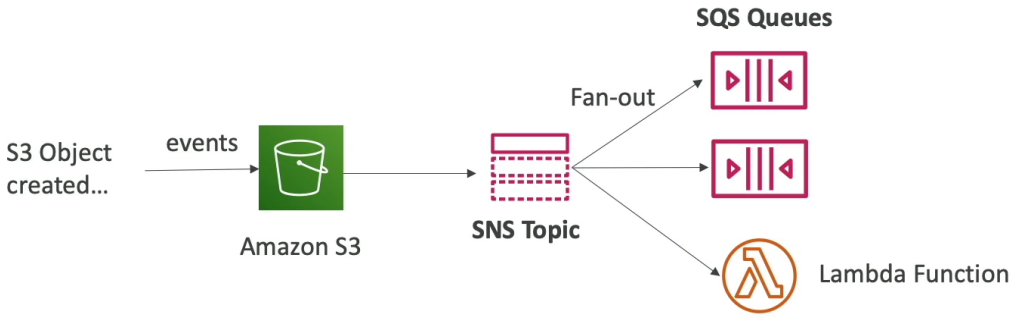
In the **Fan Out** pattern, we use SNS to broadcast messages to multiple SQS queues. Each queue can then be processed independently by separate consumers or applications. Need to configure SQS queue access policy to allow SNS to write.



It's fully decoupled and we dont have data loss. We have ability to add more SQS subscribers over time. It works with SQS queues in other regions.

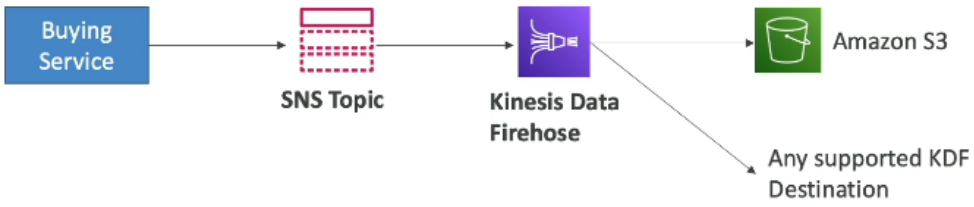
Solution Architecture - S3 Events to Multiple Queues.

For the same combination of event type and prefix we can only have one S3 Event rule. If we want to send the same S3 event to many SQS queues we should use fan out.



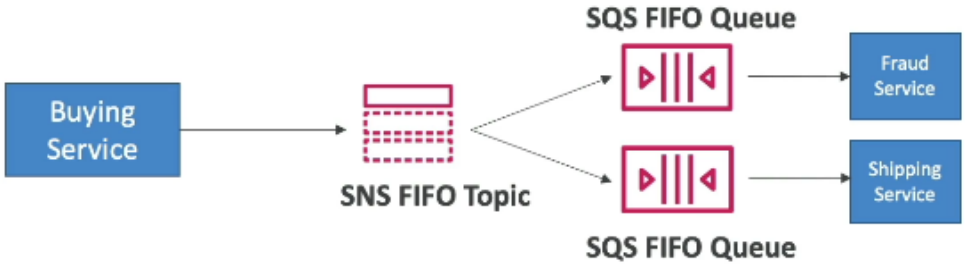
▼ **Solution Architecture - SNS to S3 through KDF.**

SNS can send to Kinesis and we can have the following solutions architecture:



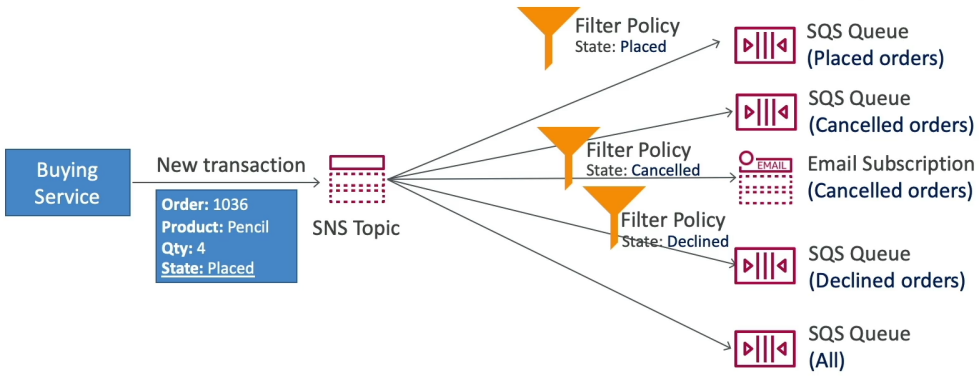
▼ **Solution Architecture - SNS FIFO + SQS FIFO Fan Out.**

In case we need fan out + ordering + deduplication.



▼ **Solution Architecture - Message Filtering.**

JSON policy is used to filter messages sent to SNS topic's subscriptions. If a subscription doesn't have a filter policy, it receives every message.



▼ ☑ **Amazon Kinesis.**

Kinesis is a managed service which collect, process and analyze real-time streaming data at any scale.

Kinesis components:

- **Kinesis Data Streams (KDS)** - collect and process large streams of data records in real time.
- **Kinesis Data Firehose** - loads streaming data into data lakes, data stores, and analytics services.
- **Kinesis Data Analytics** - analyzes streaming data using SQL or Apache Flink.
- **Kinesis Video Streams** - streams and process video in real time.

▼ ☑ **Kinesis Data Streams (KDS).**

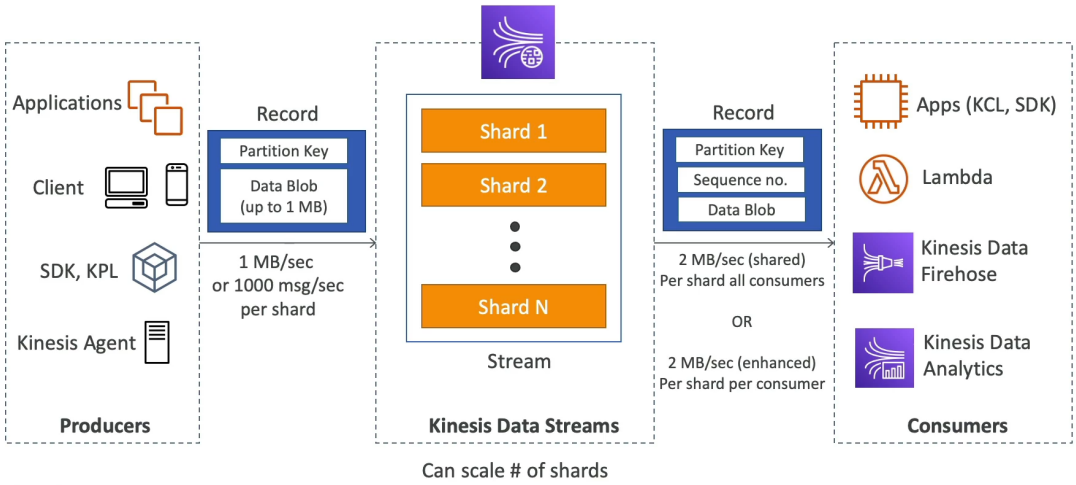
KDS is a real-time data streaming service designed to handle massive streams of data in real-time. We have ability to reprocess data. Once data is inserted in Kinesis, it cant be deleted.

Key Concepts

- **Stream** - a collection of records that are distributed across multiple shards. Each stream has a sequence of data records.
- **Shard** - the fundamental unit of capacity within a stream. A stream can have multiple shards, allowing for scaling up.
- **Record** - a unit of data in a stream. It consists of:
  - Data Blob - data we want to store.
  - Partition Key - string used to group records by shard within a stream.

    - Sequence Number - an identifier assigned to each record, unique per shard, that is used to maintain order.

- **Producers** - clients or services that write data to the KDS (AWS SDK, KPL, Kinesis Agent).
- **Consumers** - applications or services that process or consume the data (AWS SDK, KCL, AWS Lambda, KDF, KDA).



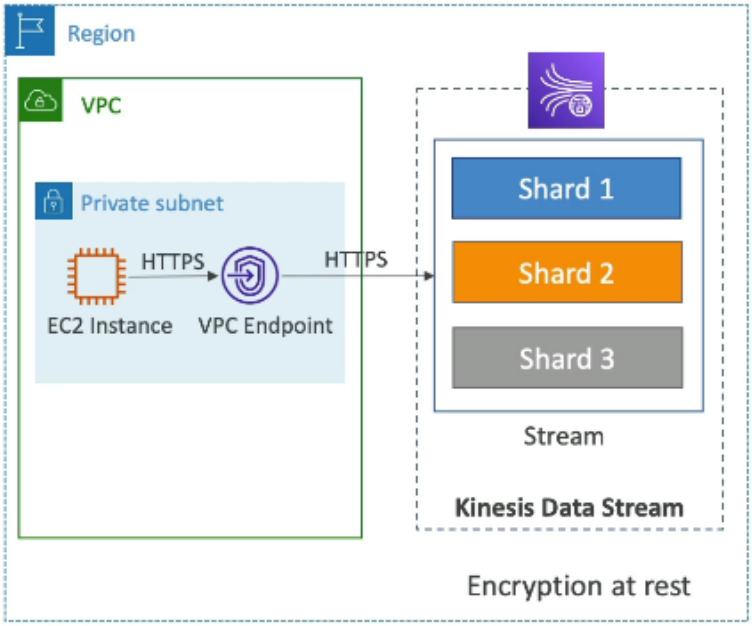
KDS Capacity Modes

- **Provisioned mode** - we choose the number of shards provisioned, scale manually or using API. We pay per shard provisioned per hour.
- **On-demand mode** - no need to provision or manage the capacity. Scales automatically based on observed throughput peak during the last 30 days. We pay per stream per hour & data in/out per GB.

KDS Security

We can control access/authorization using IAM policies. Encryption in flight is done using HTTPS endpoints and at rest using KMS. Also we can implement encryption/decryption of data on client side.

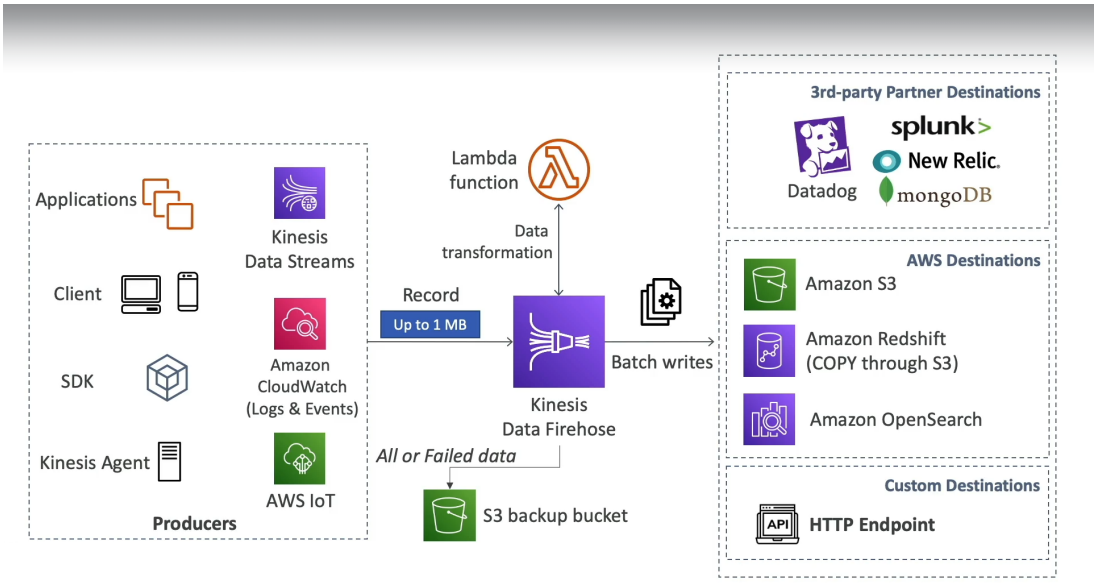
We can use VPC Endpoints for Kinesis to access within VPC. To monitor API calls we use CloudTrail.



▼ ☑ **Kinesis Data Firehose (KDF).**

KDF is a fully managed service that makes it easy to reliably load streaming data into data lakes, data stores, and analytics services. We pay for data going through Firehose.

It supports many data formats, conversions, transformations and compresion. It supports custom data transformations using AWS Lambda. Firehose can send failed or all data to a backup S3 bucket.



Because we write data in batches, KDF is near real time. If we set buffer interval to zero seconds (no buffering) it will be still considered near real time.

Kinesis Data Streams	Kinesis Data Firehose
• Streaming service for ingest at scale • Write custom code (producer / consumer) • Real-time (~200 ms) • Manage scaling (shard splitting / merging) • Data storage for 1 to 365 days • Supports replay capability	• Load streaming data into S3 / Redshift / OpenSearch / 3rd party / custom HTTP • Fully managed • Near real-time • Automatic scaling • No data storage • Doesn't support replay capability

▼ ☑ **Kinesis Data Analytics (KDA).**

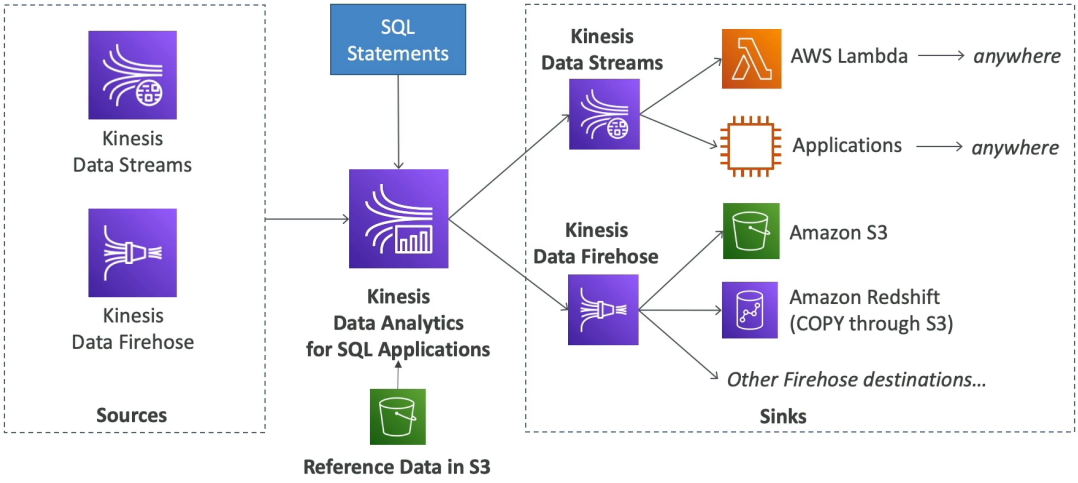
KDA for SQL Applications

It provides real-time analytics on KDS & KDF using SQL. We can add reference data from S3 to enrich streaming data. It is fully managed, no servers to provision and has automatic scaling.

Output:

- KDS - create streams out of real-time analytics query.
- KDF - send analytics query results to destinations.

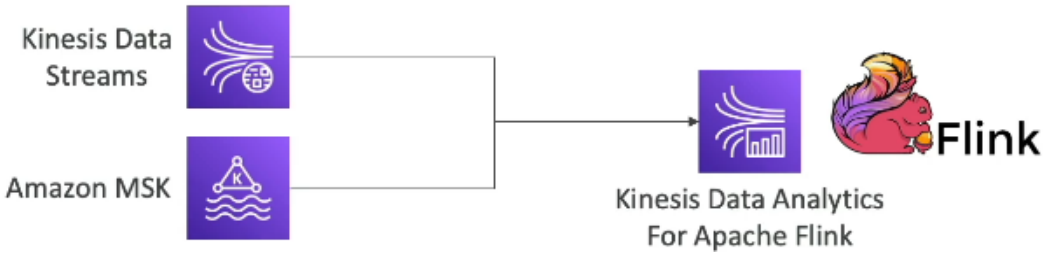
Use cases: time-series analytics, real-time dashboards, real-time metrics.



KDA for Apache Flink

Supports building and running Apache Flink applications. We get automatic provisioning compute resources, parallel computation and automatic scaling and automatic backups. Flink does not read from KDF.

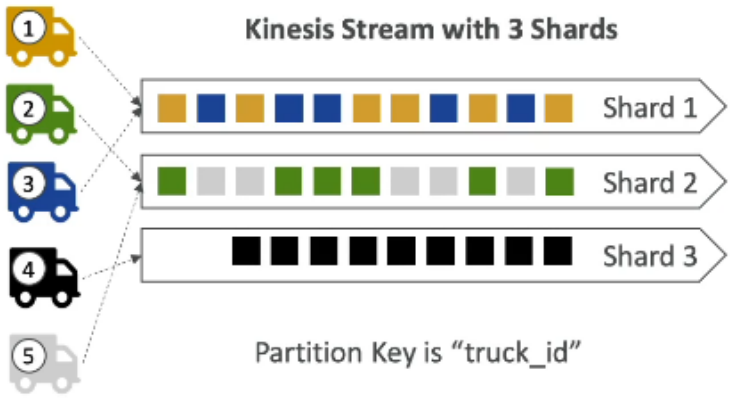
> Flink is a powerful open-source stream processing framework that provides advanced capabilities for managing state, fault tolerance, and complex event processing.



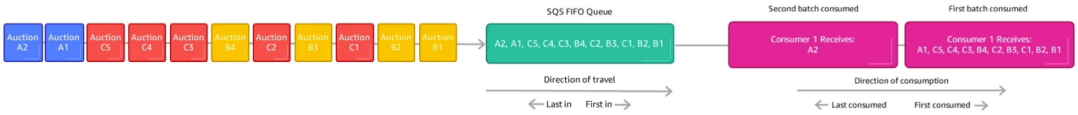
▼ ☑ **Data Ordering for KDS & SQS FIFO.**

To ensure ordering for specific types of data in KDS, we can use partition keys. Events with the same partition key are routed to the same shard, and thus will be

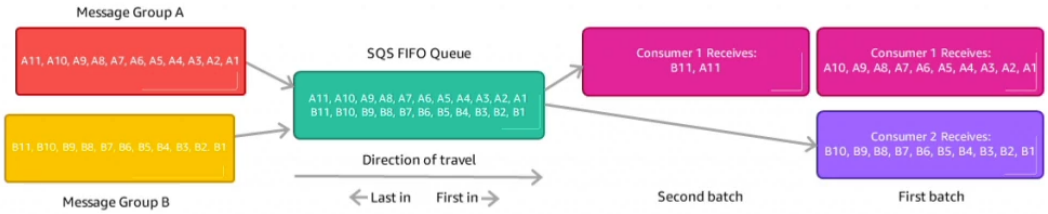
read in order.



For SQS FIFO, if we dont use a Group ID, messages are consumed in the order they are sent, with only one consumer.



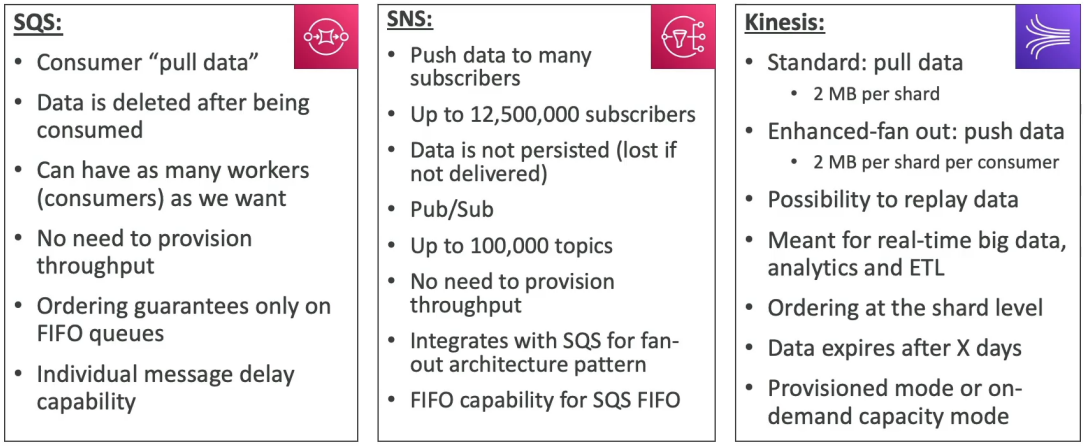
If we want to scale the number of consumers, but want messages to be “grouped” when they are related to each other, then we should use a Group ID (similar to Partition Key in Kinesis).



Lets assume 100 trucks, 5 kinesis shards, 1 SQS FIFO:

- Kinesis Data Streams:
  - On average you'll have 20 trucks per shard
  - Trucks will have their data ordered within each shard
  - The maximum amount of consumers in parallel we can have is 5
  - Can receive up to 5 MB/s of data
- SQS FIFO
  - You only have one SQS FIFO queue
  - You will have 100 Group ID
  - You can have up to 100 Consumers (due to the 100 Group ID)
  - You have up to 300 messages per second (or 3000 if using batching)

▼ ☑ SQS vs SNS vs Kinesis.



▼ ☑ AWS SWF.

SWF (Simple Workflow Service) is a fully managed service that helps us coordinate and manage complex workflows. It allows us to build and run applications that coordinate tasks across distributed components or microservices.

- **Workflow** - series of steps or tasks that need to be executed in a specific order or with specific dependencies. SWF helps us manage the execution of these workflows, ensuring that tasks are completed and handling retries and failures.
- **Activity** - task that is performed as part of the workflow. Activities are typically implemented as code that performs the work and communicates

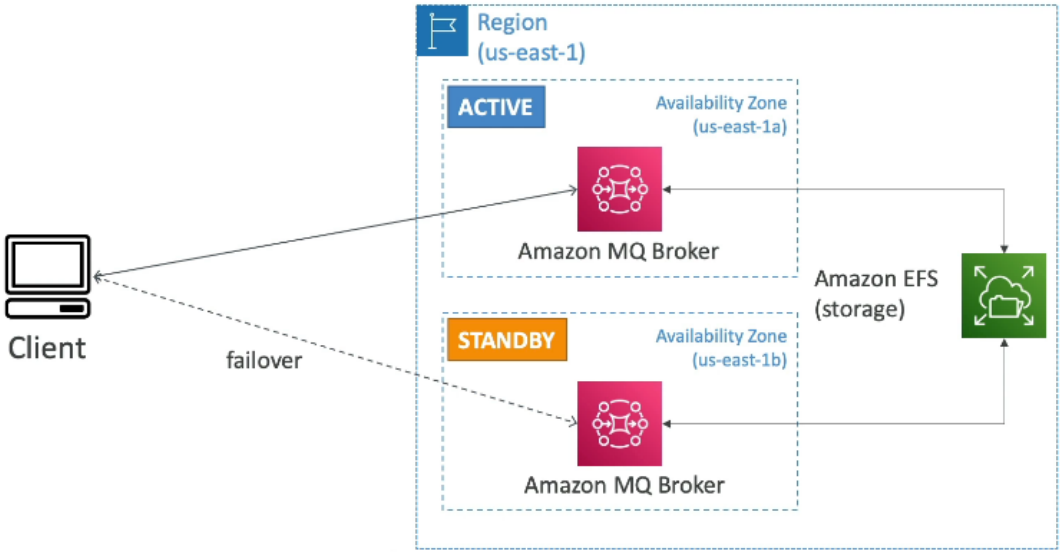
with external systems or services.

- **Task** - unit of work within an activity or a workflow. Tasks are sent to worker processes that perform the actual work.

▼ ☑ **Amazon MQ.**

When migrating to the cloud, instead of re-engineering the application to use SQS and SNS, we can use Amazon MQ. **Amazon MQ** is a managed message broker service for RabbitMQ and ActiveMQ.

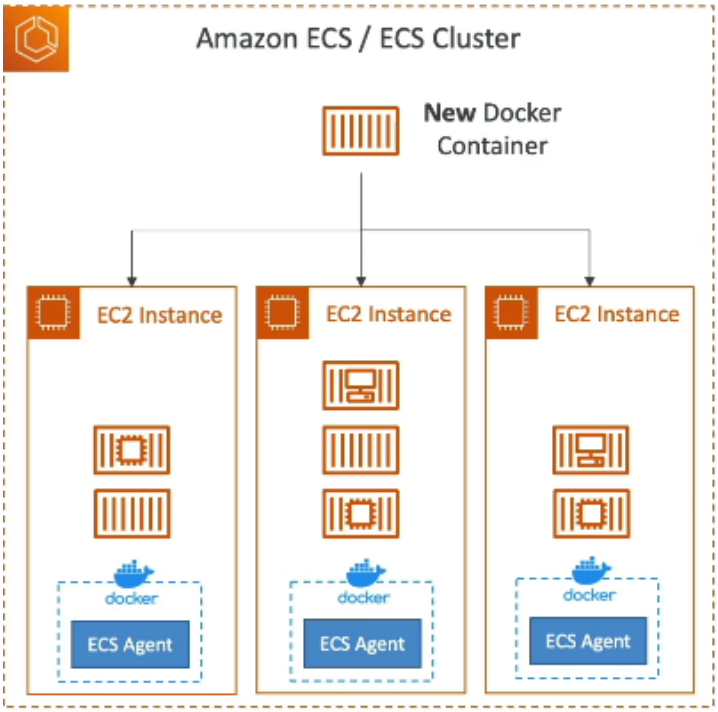
Amazon MQ doesn't scale as much as SQS and SNS. Runs on servers but, can run in Multi-AZ with failover and has both queue feature (like SQS) and topic features (like SNS).



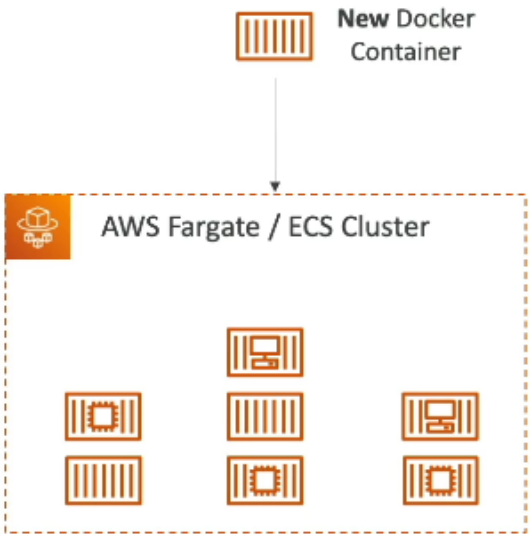
Containers on AWS

▼ ☑ **ECS & Fargate.**

ECS [EC2 Launch Type] - we must provision & maintain the infrasturcture (EC2 instances). Each EC2 Instance must run the ECS Agent to register in the ECS Cluster. AWS takes care of starting/stopping containers.

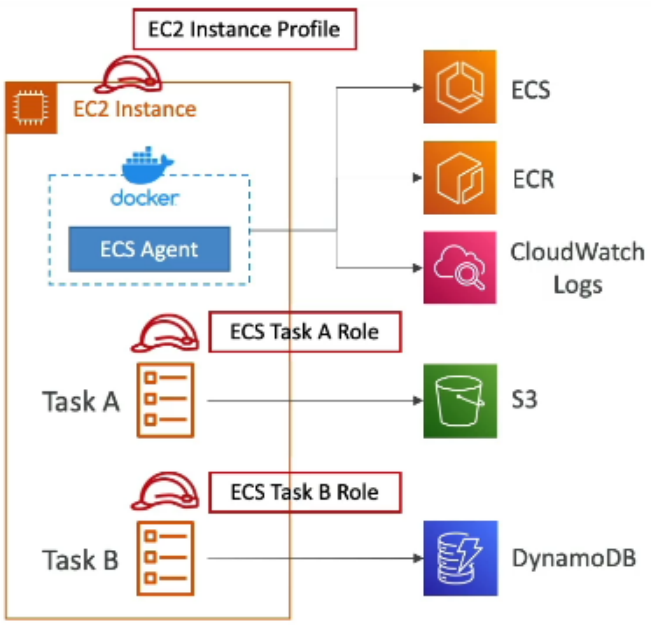


ECS [Fargate Launch Type] - we do not provision the infrasturcture, it's all serverless. We just create task definitions. AWS just runs ECS Tasks for us based on the CPU/RAM we need. To scale, we just need to increase the number of tasks (no more EC2 instances).



IAM Roles for ECS

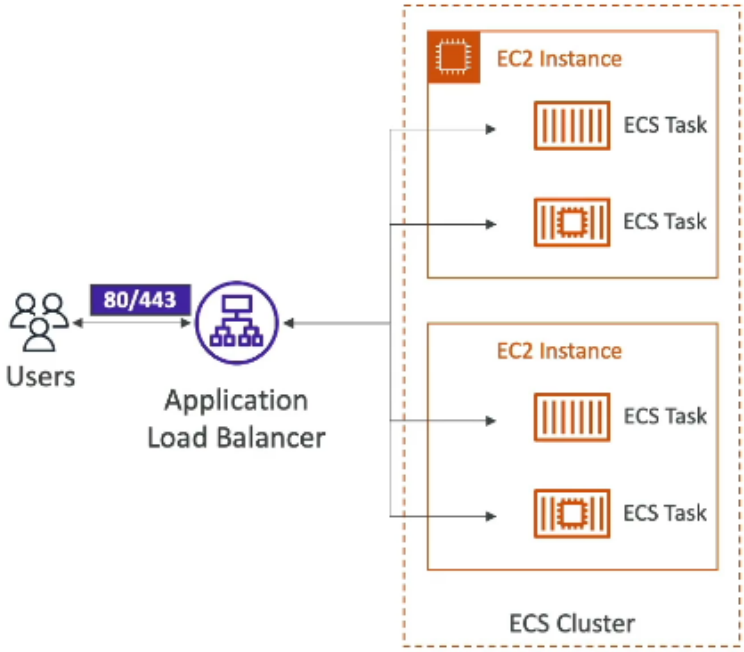
- **EC2 Instance Profile** (EC2 Launch Type only) - used by ECS agent to make API calls to ECS service. We can send container logs to CloudWatch Logs, pull Docker image from ECR, reference sensitive data in Secrets Manager or SSM Parameter Store.
- **ECS Task Role** - allows each task to have a specific role. Each role can use different ECS services.



ECS - ELB Integrations

ALB - supported and works for most use cases.

NLB - recommended only for high throughput/high performance use cases or to pair it with AWS Private Link.



ECS Data Volumes (EFS)

We can mount EFS file systems onto ECS tasks. It works for both EC2 and Fargate launch types.

Tasks running in any AZ will share the same data in the EFS file system. If we use Fargate + EFS it's totally serverless.

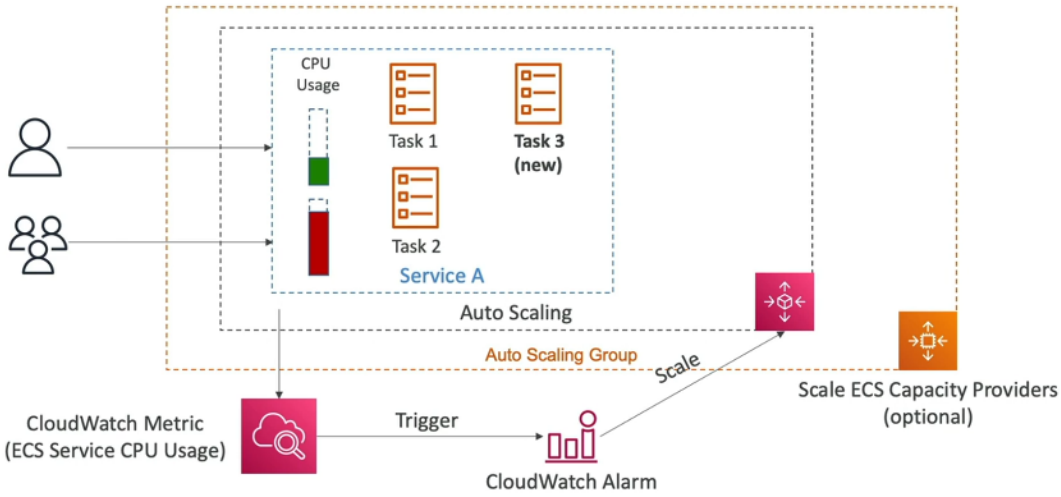
▼ ☑ **ECS Auto Scaling.**

With **ECS Auto Scaling** we can automatically increase/decrease the desired number of ECS tasks. Fargate Auto Scaling is much easier to setup because it is Serverless.

ECS Auto Scaling is similar but not equal to EC2 Auto Scaling (task level vs instance level).

- **Target Tracking Scaling** - scale based on target value for a specific CloudWatch metric.
- **Step Scaling** - scale based on a specified CloudWatch Alarm.
- **Scheduled Scaling** - scale based on a specified date/time (predictable changes).

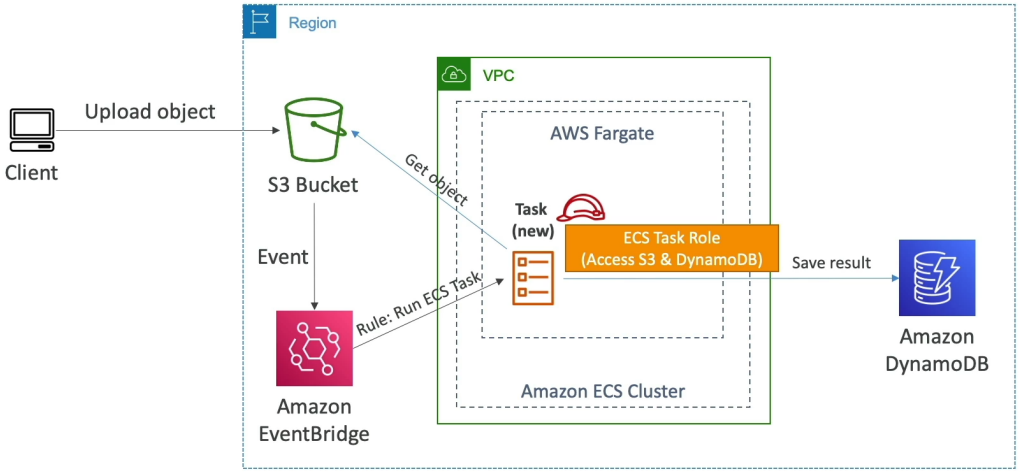
For EC2 launch type we can accommodate ECS scaling by adding underlying EC2 instances (using ASG or ECS Cluster Capacity Provider). ECS Cluster Capacity Provider is better.



▼ ☑ **ECS Solution Architectures.**

▼ **Tasks Invoked by Event Bridge.**

- Amazon EventBridge listens to events from AWS services, custom applications, or SaaS integrations.
- A rule is created in EventBridge that matches certain event patterns.
- When an event matching the rule occurs, EventBridge triggers ECS task or other service to process that event.
- For example, if an S3 object is created, an EventBridge rule can trigger a specific ECS Fargate task to process the file.



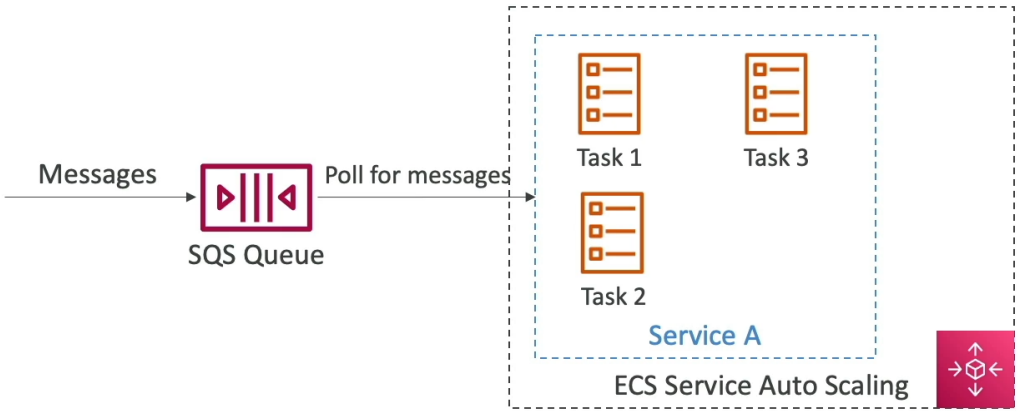
▼ Tasks Invoked by Event Bridge Schedule.

- EventBridge can be configured to run on a cron or rate expression (e.g. every hour).
- A scheduled rule in EventBridge invokes ECS tasks on the defined schedule.
- For example, we can run a scheduled ECS task every day at midnight to perform batch processing.

▼ ECS - SQS Queue.

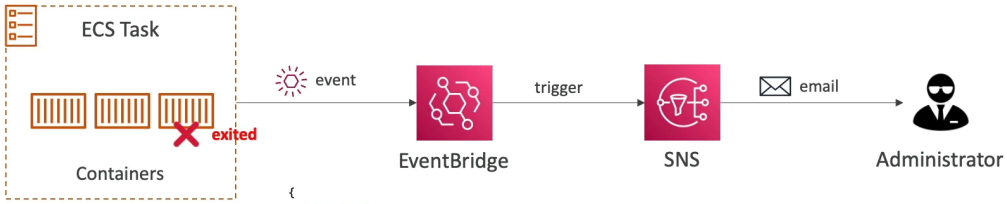
- Messages are sent to an SQS queue by producers, which can be other services, APIs, or applications.

- An ECS service with a task or set of tasks polls the SQS queue for messages.
- When a message is retrieved, the ECS task processes it. Once processed, the message is either deleted or moved to a dead-letter queue if it fails.



▼ **Intercept Stopped Tasks using Event Bridge.**

- ECS emits task state change events (e.g. task stopped, task started) which are sent to EventBridge.
- An EventBridge rule is set up to capture the “ECS Task State Change” events where the stopped status is detected.
- When a task stops, EventBridge triggers a target such as an SNS topic, Lambda function, or another ECS task for handling remediation, alerting, or cleanup actions.



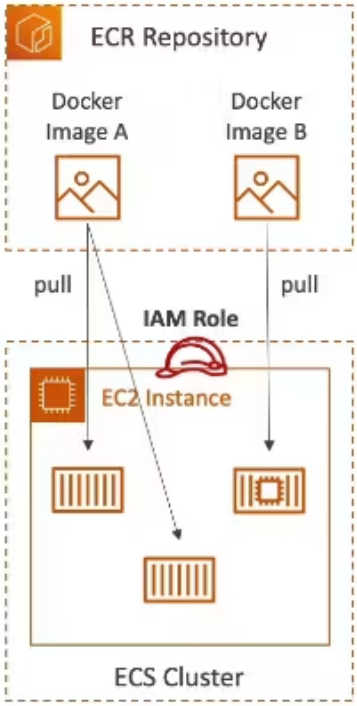

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Task State Change"
    ],
    "detail": {
        "lastStatus": [
            "STOPPED"
        ],
        "stoppedReason": [
            "Essential container in task exited"
        ]
    }
}
```

Event Pattern

▼ ☑ ECR.

ECR - private docker registry on AWS. It stores and manages Docker images on AWS. It is fully integrated with ECS and backed by S3.

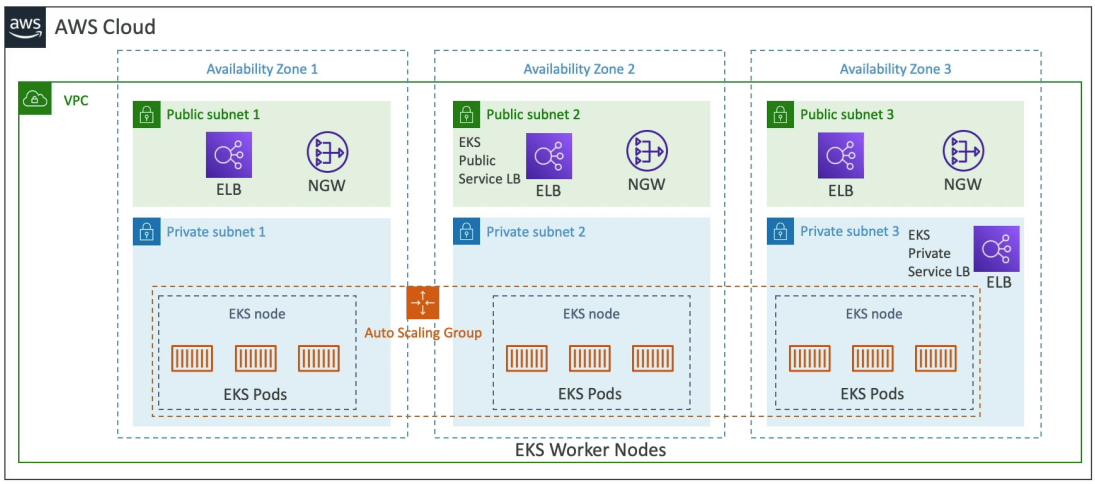
Access is controlled through IAM. It supports image vulnerability scanning, versioning, image tags, image lifecycle...



▼ ☑ EKS.

EKS is a managed service that simplifies the process of running Kubernetes on AWS. It is an alternative to ECS, similar goal but different API. Kubernetes is cloud-agnostic (can be used in any cloud).

EKS supports EC2 if we want to deploy worker nodes or Fargate to deploy serverless containers.



We can attach data volumes to our EKS cluster. We need to specify StorageClass manifest on our EKS cluster. It leverages a **Container Storage Interface (CSI)** compliant driver.

We have support for EBS, EFS (works with Fargate), FSx for Lustre, FSx for NetApp ONTAP.

Amazon EKS Connector is a feature that allows us to connect and manage Kubernetes clusters running outside of AWS (whether on-premises or in other cloud environments) within the Amazon EKS console.

Amazon EKS Anywhere is a deployment option for Amazon EKS that enables us to create and operate Kubernetes clusters on our own infrastructure, such as on-premises data centers or other cloud environments.

EKS Node Types

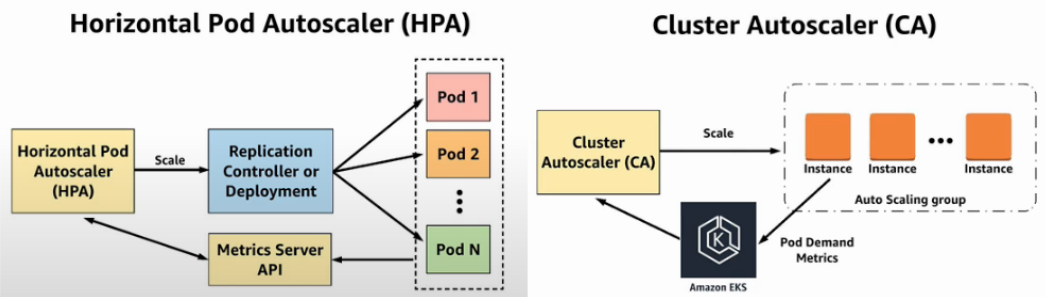
- **Managed Node Groups** - creates and manages Nodes (EC2 instances) for us. Nodes are part of an ASG managed by EKS. It supports on-demand or spot instances.

- **Self-Managed Nodes** - nodes created by us and registered to the EKS cluster and managed by an ASG. We can use prebuilt AMI (EKS Optimized AMI). It supports on-demand or spot instances.
- **AWS Fargate** - no maintenance required, no nodes managed.

▼ ☑ **EKS Auto Scaling.**

EKS Metrics Server - lightweight, scalable aggregator of resource usage data in our Kubernetes cluster. The primary use of the Metrics Server is to enable Horizontal Pod Autoscaling (HPA). HPA automatically adjusts the number of pod replicas in a deployment, replication controller, or stateful set based on observed CPU utilization.

Kubernetes Cluster Autoscaler automatically adjusts the size of the Kubernetes cluster (e.g. the number of nodes) based on the resource requirements of the workloads running on the cluster.



Karpenter - flexible, high-performance Kubernetes cluster autoscaler that launches appropriately sized compute resources, like Amazon EC2 instances, in response to changing application load. It integrates with AWS to provision compute resources that precisely match workload requirements.

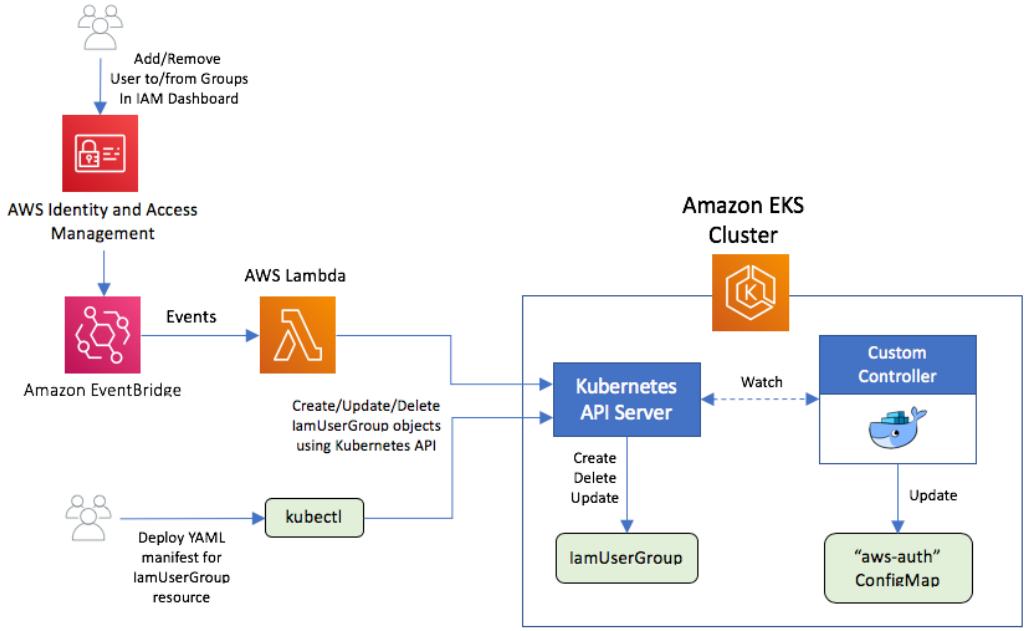
▼ ☑ **EKS Security.**

EKS uses IAM to provide authentication to our Kubernetes cluster, but it still relies on native Kubernetes Role-Based Access Control (RBAC) for authorization. This means that IAM is only used for the authentication of valid IAM entities. All permissions for interacting with our Amazon EKS cluster's Kubernetes API are managed through the native Kubernetes RBAC system.

Access to our cluster using IAM entities is enabled by the IAM Authenticator for Kubernetes, which runs on the EKS control plane. The authenticator gets its

configuration information from the *aws-auth ConfigMap* (AWS authenticator configuration map).

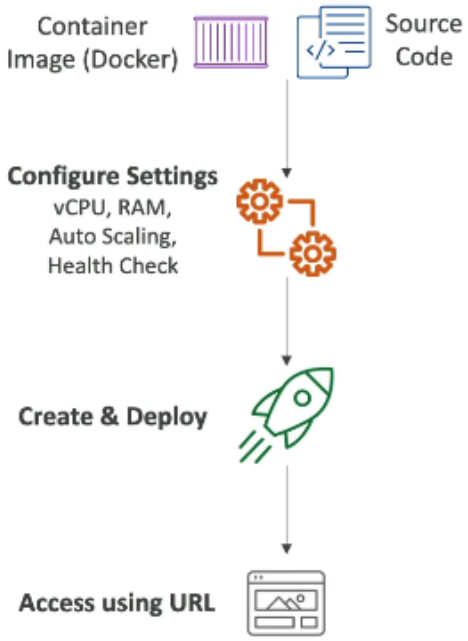
The **aws-auth ConfigMap** is automatically created and applied to our cluster when we create a managed node group or when we create a node group using **eksctl**. It is initially created to allow nodes to join our cluster, but we also use this ConfigMap to add role-based access control (RBAC) access to IAM users and roles.



▼ ☑ AWS App Runner.

App Runner is a fully managed service that makes it easy to deploy web applications and APIs at scale. It automatically builds and deploys the web app. We dont need intrastructure experience (start with our source code or container image).

We can connect to database, cache and message queue services. It has automatic scaling, load balancer and encryption.



Use cases: web apps, APIs, microservices, rapid production deployments.

AWS Serverless

▼ ☑ **AWS Lambda.**

AWS Lambda lets us run virtual functions on-demand (serverless). It has short execution and automated scaling. Functions get invoked by AWS only when needed (Event-Driven). Lambda functions are **stateless**, meaning each invocation is independent of previous invocations. State can be managed via external storage like DynamoDB or S3.

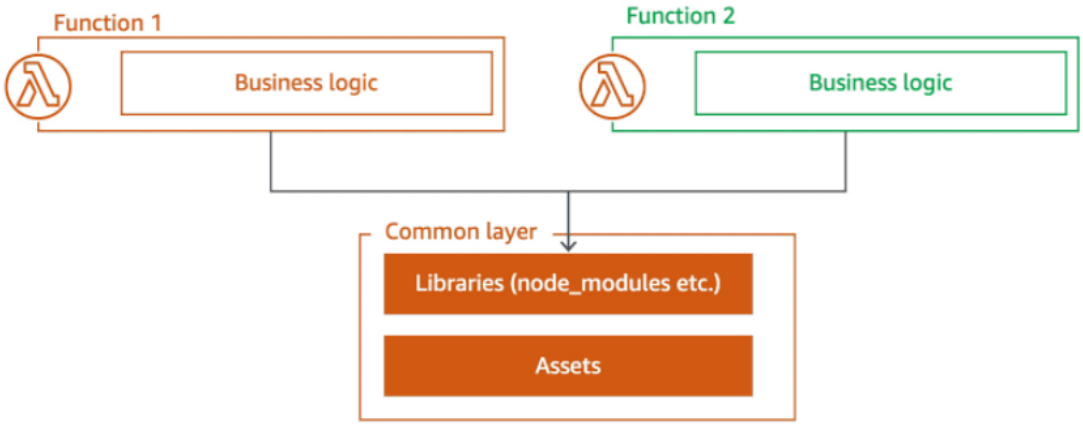
It is is integrated with the whole AWS suite of services and many programming languages. We can easily monitor AWS Lambda through AWS CloudWatch.

We **pay per request and compute time**. 1 million requests per month and 400 000 GB-seconds of compute time per month are free.

Lambda can be used for serverless CRON job (trigger an event at specific time).

Docker images can be run using Lambda. The container image must implement the Lambda Runtime API. We cant run Windows containers (only Linux).

AWS Lambda Layers - we can configure our AWS Lambda function to pull in additional code and content in the form of layers. A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. With layers, we can use libraries in our function without needing to include them in your deployment package.



AWS Lambda URLs are a feature that allows us to create HTTP endpoints for our Lambda functions. This makes it easier to build serverless applications that interact with web clients or other HTTP-based services.

Lambda Limits

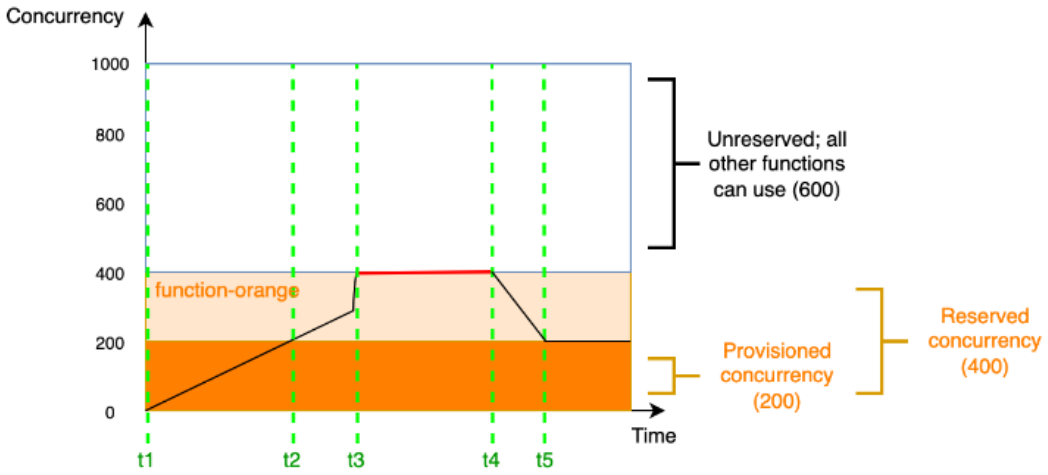
- Execution:
  - Memory allocation: 128 MB – 10GB (1 MB increments)
  - Maximum execution time: 900 seconds (15 minutes)
  - Environment variables (4 KB)
  - Disk capacity in the "function container" (in /tmp): 512 MB to 10GB
  - Concurrency executions: 1000 (can be increased)
- Deployment:
  - Lambda function deployment size (compressed .zip): 50 MB
  - Size of uncompressed deployment (code + dependencies): 250 MB
  - Can use the /tmp directory to load other files at startup
  - Size of environment variables: 4 KB

Since AWS Lambda functions can scale extremely quickly, it's a good idea to deploy a Amazon CloudWatch Alarm that notifies our team when function metrics such as ConcurrentExecutions or Invocations exceeds the expected threshold.

▼ ☑ **AWS Lambda Scaling.**

We can configure number of Lambda function which will execute simultaneously. Lambda automatically scales our function by running multiple instances of our code in parallel to handle high volumes of requests.

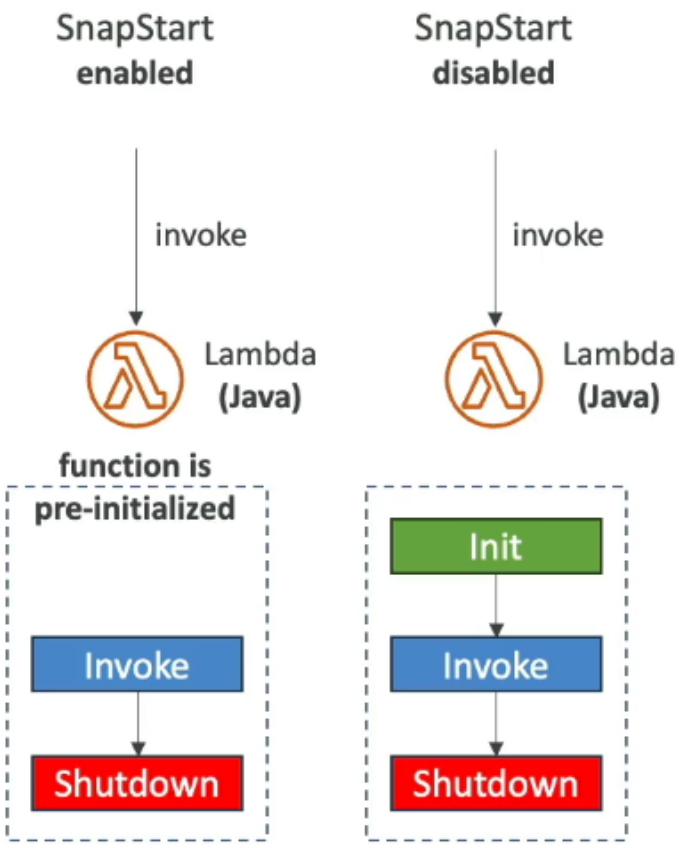
- **Provisioned Concurrency** - specified number of function instances are **pre-warmed** and ready to handle requests, which helps reduce latency and avoid cold starts. It's useful for functions with predictable traffic patterns or **latency-sensitive** applications.
- **Reserved Concurrency** - allows us to allocate a specific number of concurrent executions for a Lambda function, which ensures that the function always has the specified number of concurrent executions available and controls its maximum concurrency.



▼ ☑ **AWS Lambda SnapStart.**

Lambda SnapStart improves our Lambda functions performance up to 10x at **no extra cost** for Java 11 and above. When enabled, function is invoked from a preinitialized state (no function initialization from scratch).

When we publish a new version, Lambda initializes our function, takes a snapshot of memory and disk state of the initialized function and then snapshot is cached for low-latency access.



Lambda Invocation Lifecycle Phases

▼ ☐ **Edge Functions.**

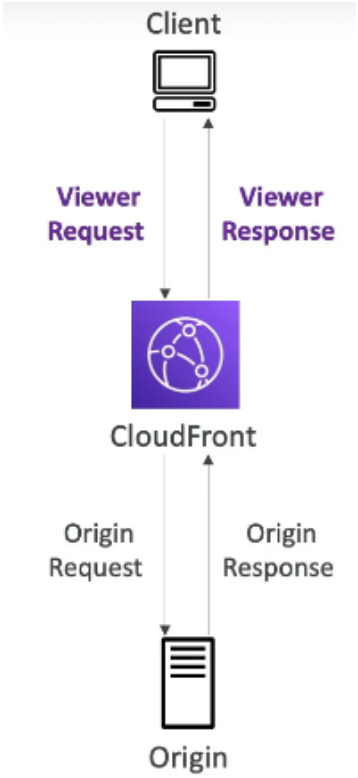
Edge Function - a code that we write and attach to CloudFront distributions. It runs close to our users to minimize latency. CloudFront provides two types: **CloudFront Functions** & **Lambda@Edge**. We dont have to manage any servers (fully serverles, pay only for what we use) and it is deployed globally.

Use cases: website security, dynamic web app at the edge, SEO (Search Engine Optimization), bot mitigation at the edge, real-time image transformation ...

CloudFront Functions - lightweight functions written in JavaScript. Used for high-scale, latency-sensitive CDN customizations (sub-ms startup times, millions of requests/second).

They are used to change viewer requests and responses.

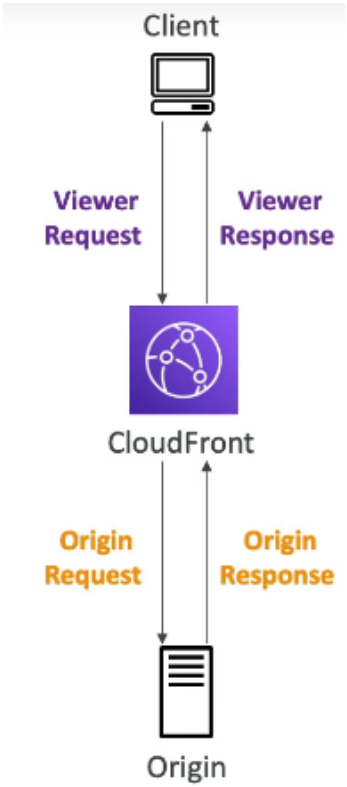
- Viewer Request - after CloudFront receives a request from a viewer.
- Viewer Response - before CloudFront forwards the response to the viewer.



Lambda@Edge - Lambda functions written in NodeJS or Python. It scales to 1000s of requests/seconds. They are used to change CloudFront requests and responses.

- Origin Request - before CloudFront forwards the request to the origin.
- Origin Response - after CloudFront receives the response from the origin.

Author our functions in one AWS Region (us-east-1), then CloudFront replicates to its locations.



CloudFront Functions vs Lambda@Edge

	CloudFront Functions	Lambda@Edge
Runtime Support	JavaScript	Node.js, Python
# of Requests	**Millions** of requests per second	**Thousands** of requests per second
CloudFront Triggers	- Viewer Request/Response	- Viewer Request/Response - Origin Request/Response
Max. Execution Time	< 1 ms	5 – 10 seconds
Max. Memory	2 MB	128 MB up to 10 GB
Total Package Size	10 KB	1 MB – 50 MB
Network Access, File System Access	No	Yes
Access to the Request Body	No	Yes
Pricing	Free tier available, 1/6th price of @Edge	No free tier, charged per request & duration

CloudFront Functions

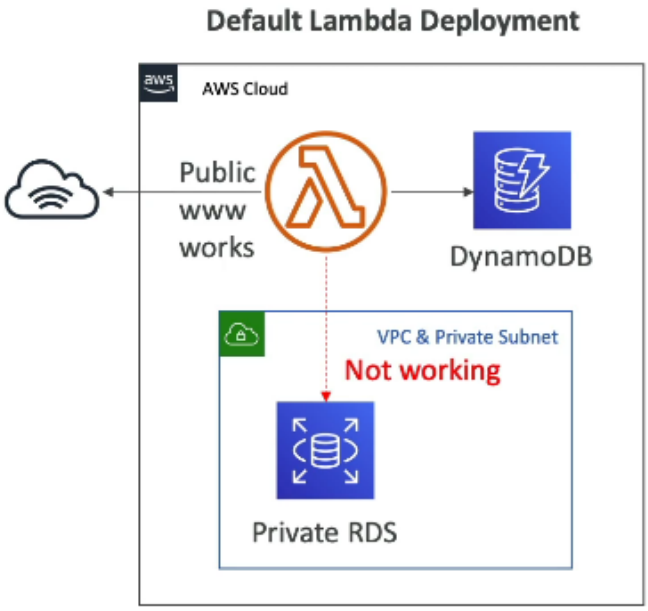
- Cache key normalization
  - Transform request attributes (headers, cookies, query strings, URL) to create an optimal Cache Key
- Header manipulation
  - Insert/modify/delete HTTP headers in the request or response
- URL rewrites or redirects
- Request authentication & authorization
  - Create and validate user-generated tokens (e.g., JWT) to allow/deny requests

Lambda@Edge

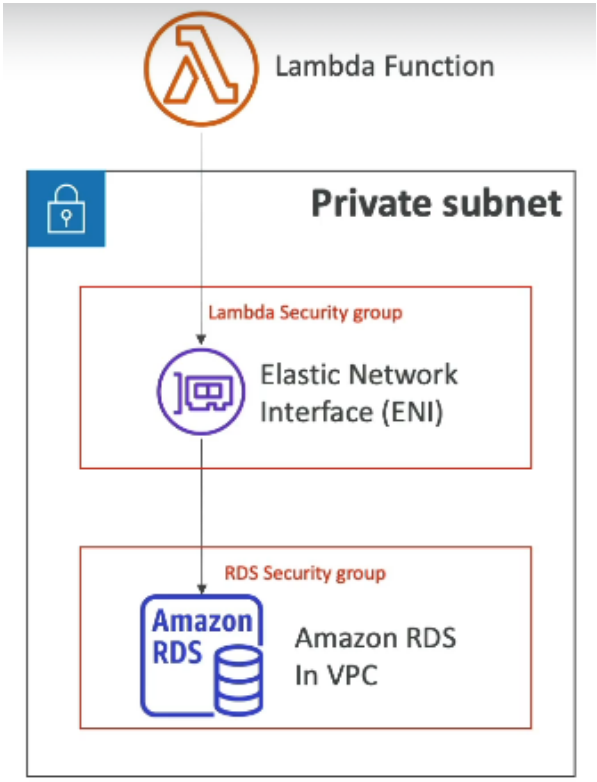
- Longer execution time (several ms)
- Adjustable CPU or memory
- Your code depends on a 3rd libraries (e.g., AWS SDK to access other AWS services)
- Network access to use external services for processing
- File system access or access to the body of HTTP requests

▼ ☑ **Lambda in VPC.**

By default, our Lambda function is launched outside our own VPC. Therefore, it cannot access resources in our VPC (RDS, ElastiCache, etc).



We must define the VPC ID, the subnets and security groups. Lambda will create an ENI in our subnets.

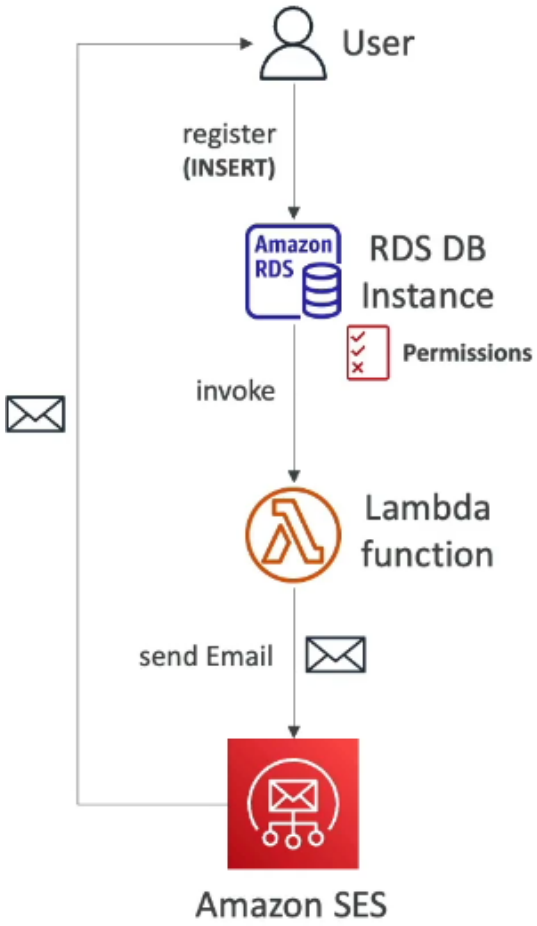


If Lambda functions directly access our database, they may open too many connections under high load. We can use RDS Proxy to improve scalability. The Lambda function must be deployed in our VPC, because RDS Proxy is never publicly accessible.

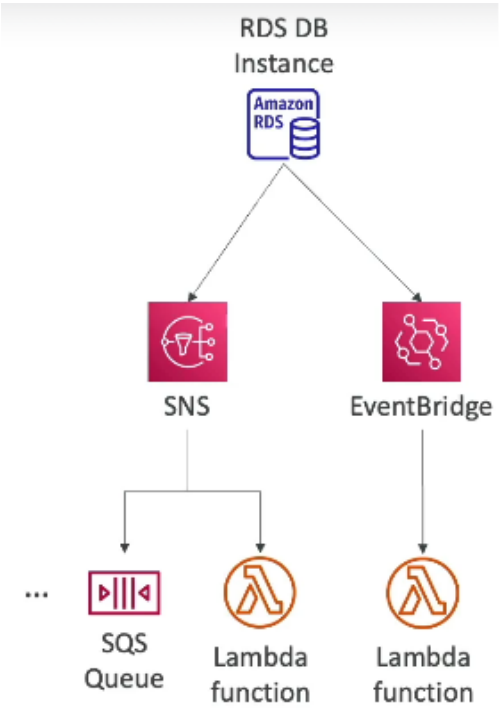
▼ ☐ **RDS - Invoking Lambda & Event Notifications.**

We can invoke Lambda functions from within our DB instance. It allows us to process data events from within a database. It's **supported for RDS for Postrgres and Aurora MySQL**.

We must allow outbound traffic to our Lambda function from within our DB instance (Public, NAT Gateway, VPC Endpoints). DB instance must have the required permissions to invoke the Lambda.



Event Notifications - notifications that tell information about the DB instance itself. It is near real-time event. We dont have any information about the data itself. Can send notifications to SNS or subscribe to events using EventBridge.

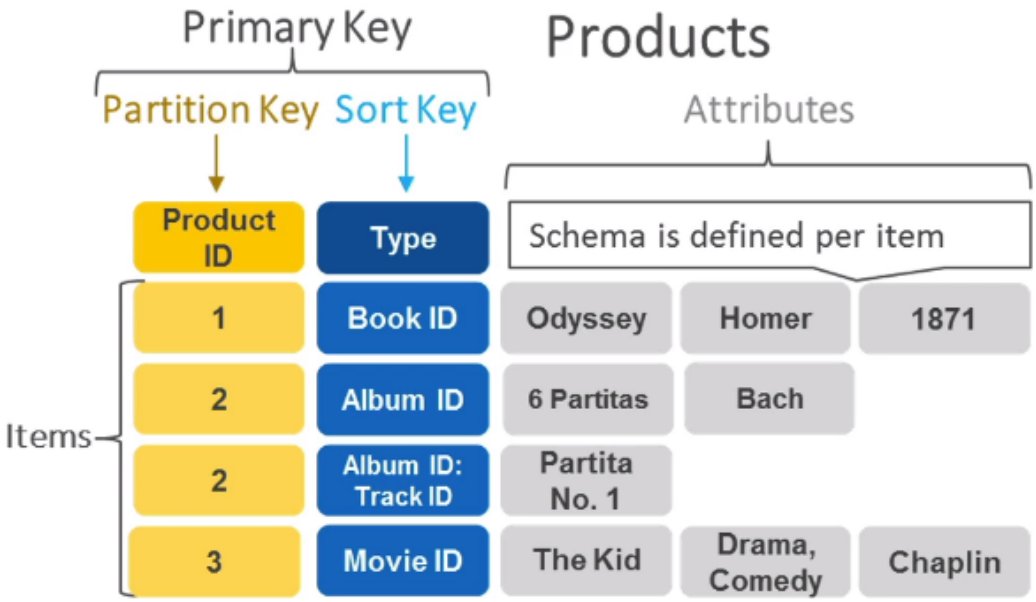


Need to subscribe to the following event categories: DB instance, DB snapshost, DB Parameter Group, DB Security Group, RDS Proxy, Custom Engine Version.

▼ ☑ **DynamoDB.**

DynamoDB is fully managed and highly available NoSQL schemaless database with replication across 3 AZ. It's distributed serverless database which can scale to massive workloads. DynamoDB is fast and consisent in performance (can handle millions of requests per second, trillions of row and 100s of TB of storage). It's low cost and has auto scaling capabilities. It can rapidly evolve schemas. It has **single-digit millisecond latency** (low latency retrieval).

DynamoDB is integrated with IAM for security, authorization and administration.



DynamoDB uses the **partition key** value as input to an internal hash function. The output from the hash function determines the partition in which the item will be stored. All items with the same partition key are stored together, in sorted order by sort key value. If no sort key is used, no two items can have the same partition key value.

Read/Write Capacity Modes

- **Provisioned Mode** (default) - we specify the number of reads/writes per second and need to plan capacity beforehand. We pay for provisioned Read Capacity Units (RCU) & Write Capacity Units (WCU). It is possible to add auto scaling mode for RCU & WCU.

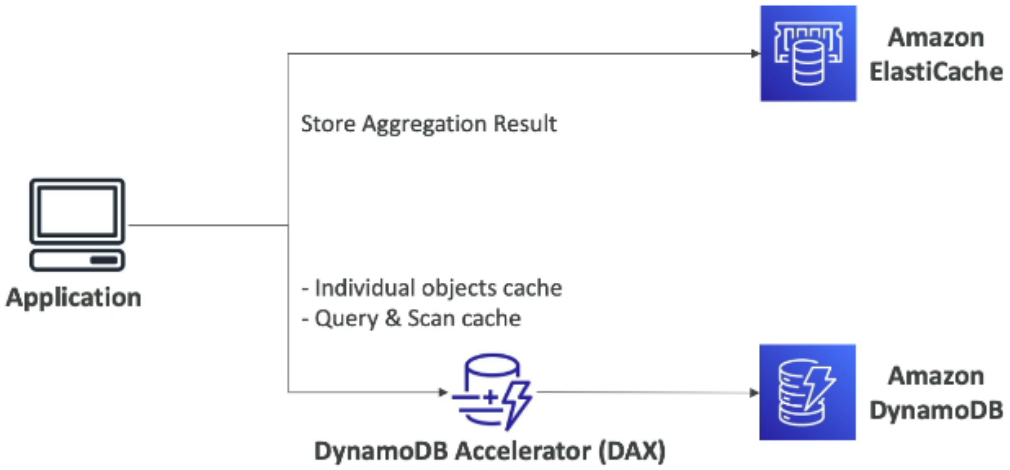
  The more distinct partition key values that our workload accesses, the more those requests will be spread across the partitioned space.

- **On-demand Mode** - read/writes automatically scale up/down with our workloads. No capacity planning is needed. We pay for what we use (more expensive). It is great for unpredictable workloads, steep sudden spikes.

▼ ☑ **DynamoDB Advanced Features.**

DynamoDB Accelerator (DAX) - fully managed, secure, highly scalable and highly available in-memory cache for DynamoDB. It gives 10x performance

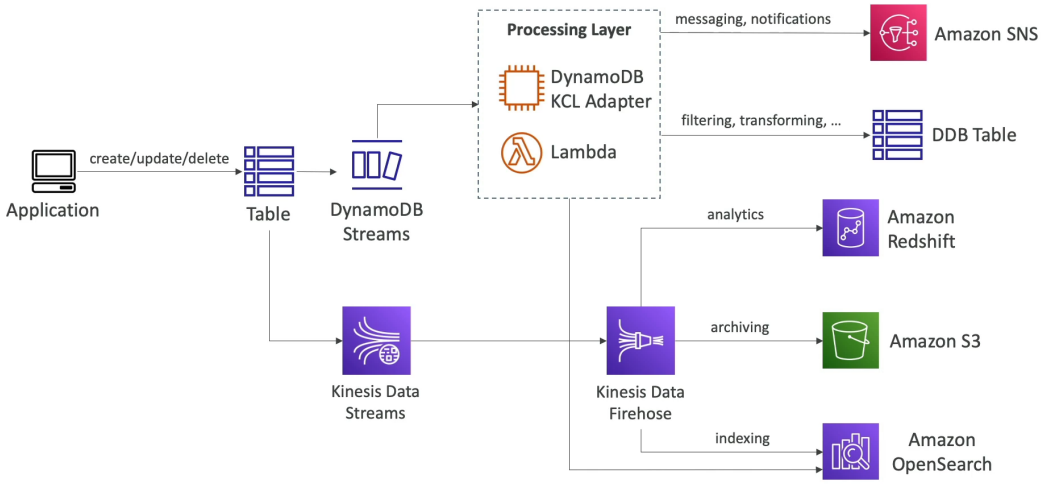
improvement (single digit millisecond latency to microseconds latency). It doesnt require application logic modification (compatible with existing DynamoDB APIs). It has 5 minutes TTL for cache (default).



DynamoDB Stream Processing - ordered stream of item-level modifications (create/update/delete) in a table.

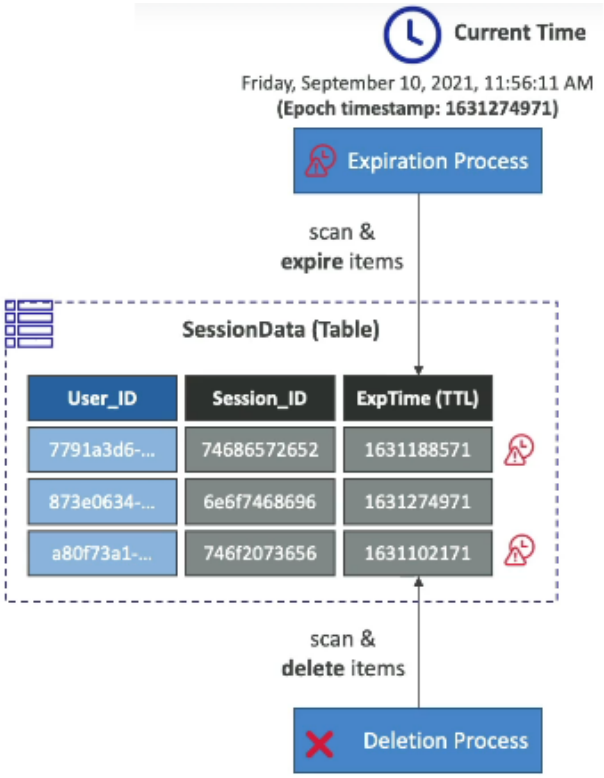
It is used for reacting to changes in real time, real time usage analytics, implementing cross-region replication, invoke AWS Lambda on changes to our DynamoDB table...

- **DynamoDB Streams** - process using AWS Lambda Triggers or DynamoDB Stream Kinesis adapter. We have 24h retention and it has limited number of consumers.
- **KDS** - Process using AWS Lambda, KDA, KDF, AWS Glue Streaming ETL... We have 1 year retention and it has high number of consumers.



DynamoDB Global Tables - feature that makes DynamoDB table accessible with low latency in multiple regions. With this we can read and write to any AWS Region and it is called Active-Active Replication. We must enable DynamoDB Streams as a pre-requisite.

DynamoDB TTL - automatically delete items after an expiry timestamp. Used to reduce stored data by keeping only current items, adhere to regulatory obligations, web session handling...



DynamoDB Backups

- **Continuous backups using PITR** - optionally enabled for the last 35 days. PITR to any time within the backup window. The recovery process creates a new table.
- **On-demand backups** - full backups for long-term retention, until explicitely delted. It doesnt affect performance or latency. Can be configured and managed in AWS Backup. The recovery process creates a new table.

DynamoDB Integration with S3

- **Export to S3** (**must enable PITR**) - works for any point of time in the last 35 days. It doesnt affect the read capacity of our table. We can perform data analysis on top of DynamoDB and ETL on top of S3 data before importing back to DynamoDB.
- **Import to S3** - import CSV, DynamoDB JSON or ION format. It doesnt consume any write capacity. It creates a new table. Import errors are logged in CloudWatch Logs.

▼ ☑ API Gateway.

Amazon API Gateway is a fully managed service for developers to easily create, publish, maintain, monitor and secure APIs. It is serverless and scalable, supports RESTful APIs and WebSocket APIs. Has support for security, user authentication, API keys... We can validate requests and responses (also cache API responses). We pay only for the API calls we receive and the amount of data transferred out.

We can monitor API usage, latency, and error rates using CloudWatch.

API Gateway Integrations

- **Lambda Function** - can invoke AWS Lambda functions within our account and start AWS Step Functions state machines.
- **HTTP** - can make HTTP requests to endpoints hosted on AWS services such as Elastic Beanstalk and EC2, as well as to non-AWS HTTP endpoints accessible over the public internet.
- **AWS Services** - can be directly integrated with other AWS services. For example, we can configure an API method to send data directly to Amazon Kinesis.

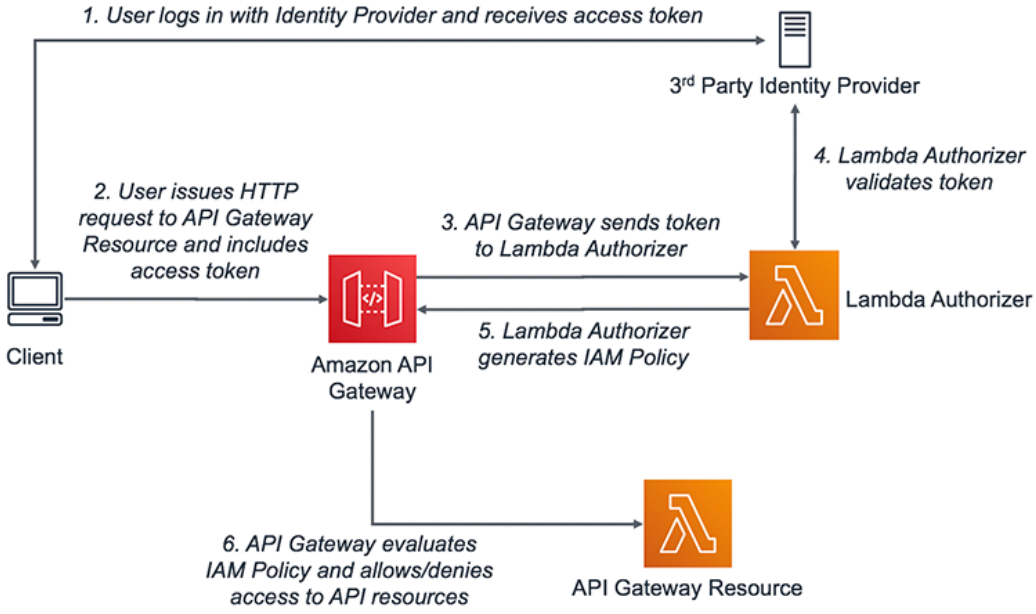
API Gateway Endpoint Types

- **Edge-Optimized** (default) - for global clients. Requests are routed through the CloudFront Edge locations. The API Gateway still lives in only one region.
- **Regional** - for clients within the same region. Cloud manually combine with CloudFront (more control over the caching strategies and the distribution).
- **Private** - can only be accessed from our VPC using an interface VPC endpoint (ENI). We need to use a resource policy to define access.

API Gateway Security

User authentication through IAM roles (for internal apps), cognito (for external users) or custom authorizer (our own logic).

Lambda Authorizer (Custom Authorizer) is a feature of AWS API Gateway that allows us to control access to our APIs using custom authentication and authorization logic implemented in AWS Lambda functions.



Custom Domain Name HTTPS security through integration with ACM. If using Edge-Optimized endpoint, then the certificate must be in us-east-1. If using Regional endpoint, the certificate must be in the API Gateway region. We must setup CNAME or Alias record in Route 53.

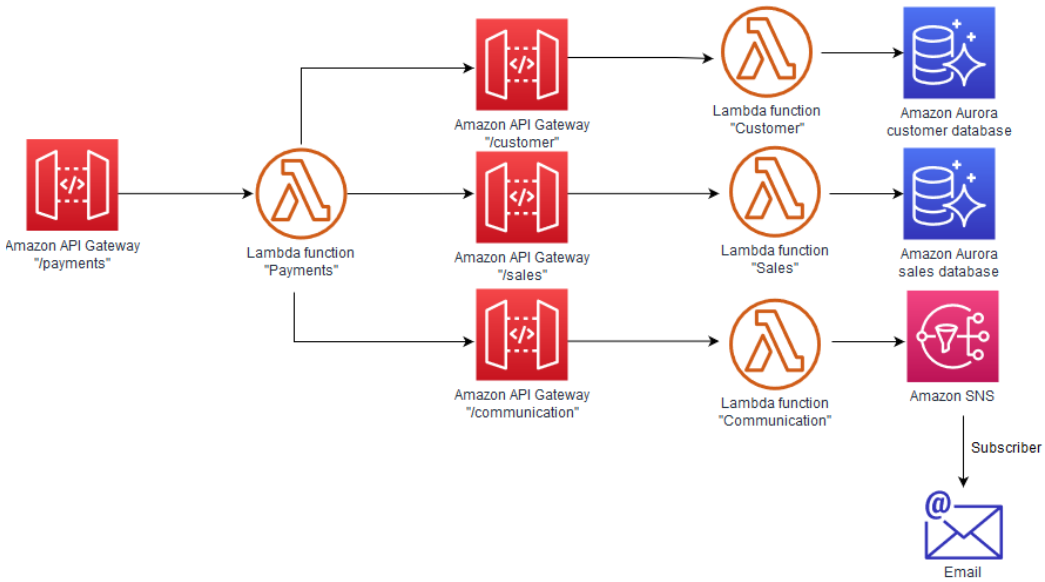
▼ ☑ **API Gateway Advanced Features.**

API Usage Plan defines who can access our API, how much they can access, and how the usage is monitored and throttled. It helps us manage and control API traffic, set quotas, and monitor usage metrics.

Custom Domain Names allows us to use our own domain names for our APIs, providing a more professional and user-friendly experience. We can map multiple APIs to different paths on a single domain using base path mappings.

▼ **Solution Architecture - Serverless Microservice Architecture.**

We can break down applications into smaller, loosely coupled services that are independently deployable and scalable without managing servers. It scales automatically and we only pay for what you use.

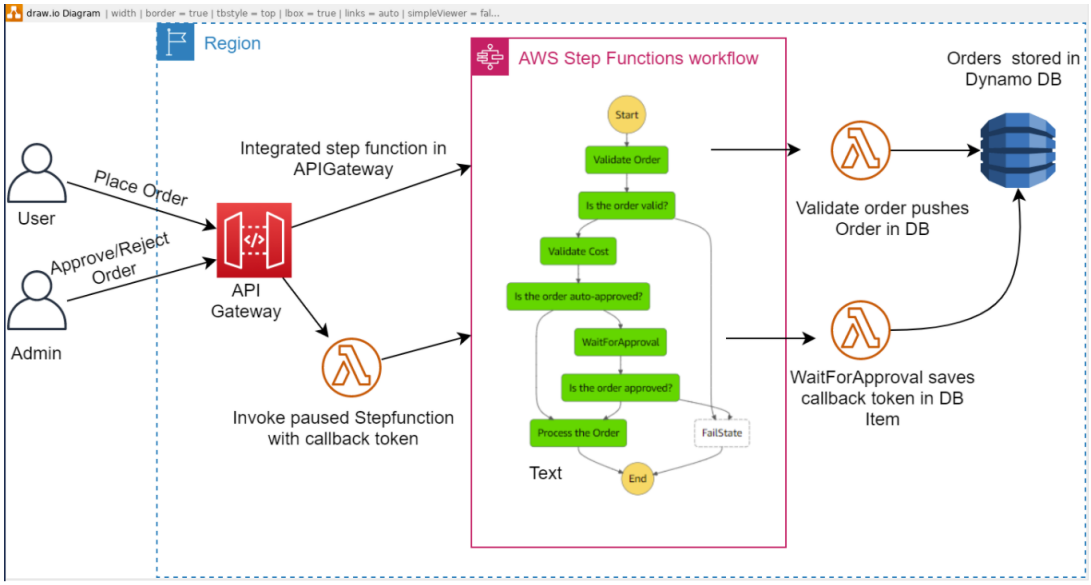


▼ ☑ AWS Step Functions.

Step Functions is a serverless orchestration service that allows us to coordinate multiple AWS services into serverless workflows. It orchtestrates our Lambda functions.

It can be integrated with EC2, ECS, on-remises servers, API Gateway, SQS, etc ...

We can use it for implementing human approval feature.



Step Functions Local is a tool that allows us to run and test our Step Functions state machines locally on our development machine. It can help us develop and debug your workflows without needing to deploy them to the AWS cloud, which can save time and costs.

▼ ☐ AWS Step Functions Map State

AWS Step Functions' Map State is a feature designed to iterate over a collection of items and process each one in parallel or sequentially, depending on how we configure it. It allows us to specify a JSON array (or list) as input and then iterate over that array and apply a specified state machine (or sub-state machine) to each item.

The Map state allows us to configure how many iterations are processed in parallel. We can set a MaxConcurrency value to limit the number of parallel executions.

- **Inline Mode** - each item in the array is processed using the same state machine specified directly within the Map state's Iterator. State machine logic defined in the Iterator is directly applied to each item in the array.

```
{
  "Type": "Map",
  "ItemsPath": "$.items",
  "Iterator": {
    "StartAt": "ProcessItem",
    "States": {
      "ProcessItem": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:REGION:ID:function:Proc
        "End": true
      }
    }
  },
  "End": true
}
```

- **Distributed Mode** - the Map state is used in conjunction with a nested state machine or sub-state machine that is applied to each item in the array. It

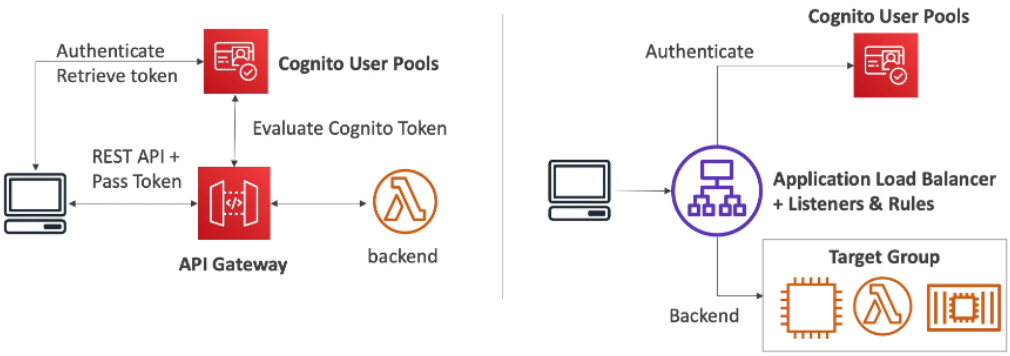
allows a more complex or different workflow to be used for each item. It is designed specifically for **massive parallelism** by breaking up large-scale processing tasks into smaller batches.

```
{
  "Type": "Map",
  "ItemsPath": "$.items",
  "Iterator": {
    "StartAt": "SubMachine",
    "States": {
      "SubStateMachine": {
        "Type": "StateMachine",
        "StateMachineArn": "arn:aws:states:REGION:ID:stateM
        "End": true
      }
    }
  },
  "End": true
}
```

▼ ☑ **Amazon Cognito.**

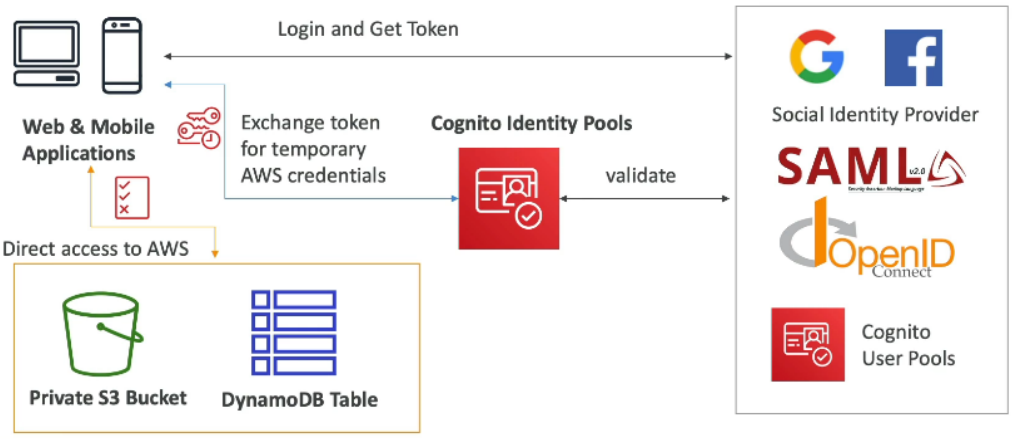
Cognito gives users an identity to interact with our web or mobile applications.

- **Cognito User Pools (CUP)** - serverless database of user for our web & mobile apps. It provides sign in functionality for app users. Integration with **API Gateway** & **ALB**. It provides simple login (username/mail and password combination), password reset, 2FA, MFA...



- **Cognito Identity Pools (Federated Identity)** - used to get identities for "users" so they obtain temporary AWS credentials. Users source can be CUP, third party logins, etc...

  Users can then access AWS services directly or through API Gateway. The IAM policies applied to the credentials are defined in Cognito. There are default IAM roles for authenticated and guest users.



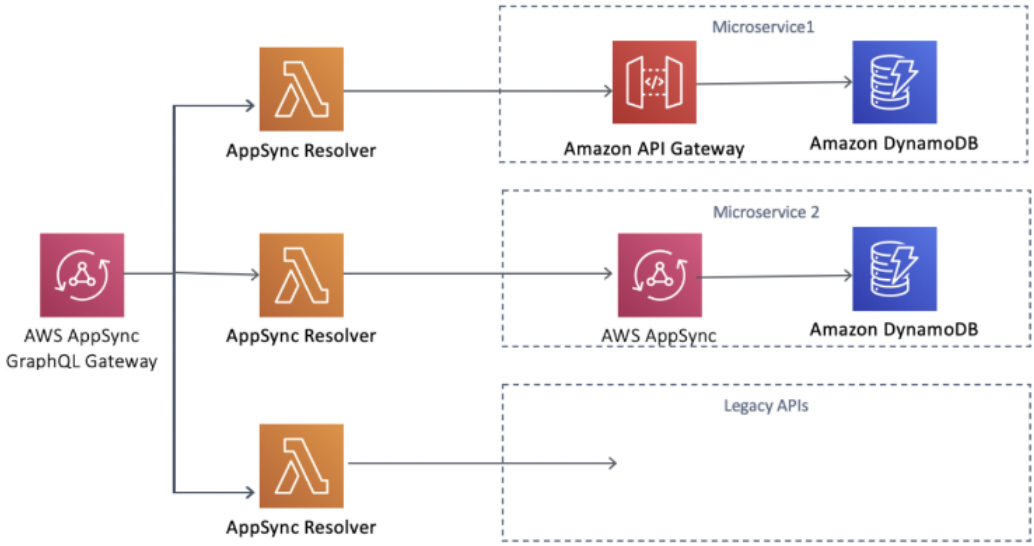
Row Level Security in DynamoDB

▼ ☐ **AWS AppSync.**

AWS AppSync is a fully managed service that simplifies the development of GraphQL and real-time APIs by enabling us to build scalable applications with ease. It **acts as a bridge** between our clients (such as web or mobile apps) and our data sources (like DynamoDB, Lambda, RDS, and other AWS services). It manages the complex backend work such as authentication, **caching**, data synchronization, and access control.

AppSync provides real-time updates to connected clients, making it ideal for use cases requiring instant data synchronization.

- **AppSync Resolvers** - determine how GraphQL queries or mutations should fetch or manipulate data.
- **AppSync Mapping Templates** - transform incoming requests into the appropriate format for the backend and then return the response.

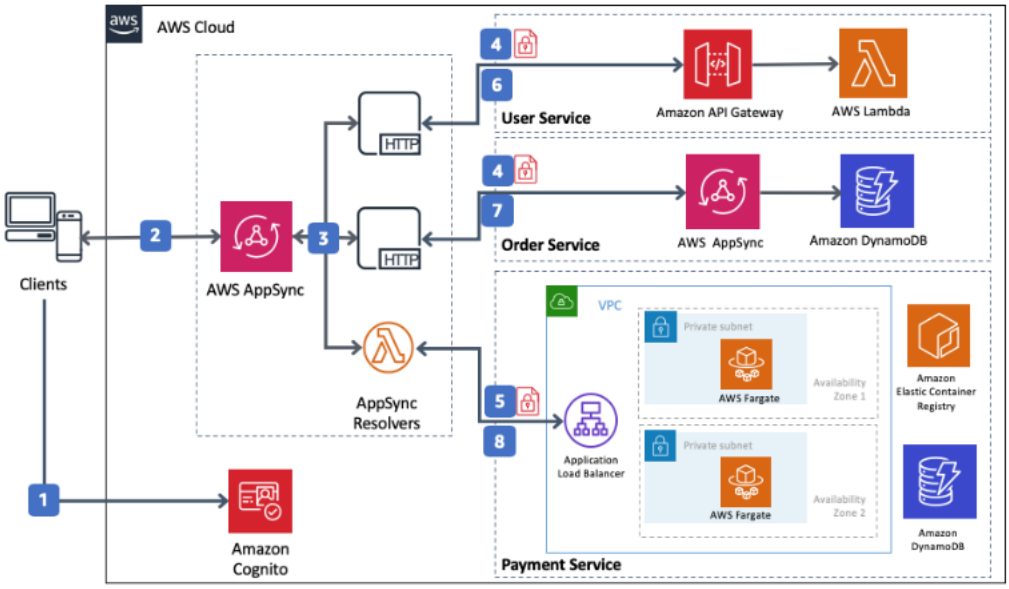


AppSync Security

AppSync integrates with multiple AWS authentication providers, such as **Amazon Cognito**, API keys, IAM, or OpenID Connect (OIDC). We can implement fine-grained access control using role-based permissions and authorization rules.

Field-level Authorization - we can restrict access down to specific fields in our GraphQL schema, ensuring only authorized users can access sensitive data.

▼ **Solution Architecture - Access to Multiple Microservices.**



Section III

Databases in AWS

▼ ☑ **RDS Summary.**

- Managed Postgres/MySQL/Oracle/SQL Server/DB2/MariaDB/Custom.
- Provisioned RDS instance size & EBS Volume type & size.
- Auto scaling capability for storage.
- Support for Read Replicas & Multi-AZ.
- Security through IAM, Security Groups, KMS, SSL in transit.
- Automated backup with PITR (35 days).
- Manual DB Snapshots for longer-term recovery.
- Managed and scheduled maintenance (with downtime).
- Support for IAM authentication, integration with Secrets Manager.
- RDS Custom for access to and customize the underlying instance (Oracle & SQL Server).
- Used for RDBMS & OLTP.

▼ ☑ **Aurora Summary.**

- Compatible API for Postgres & MySQL, separation of storage and compute.
- Data is stored in 6 replicas, across 3 AZ - highly available, self-healing, auto scaling.
- Cluster of DB instance across multiple AZ, auto scaling of Read Replicas.
- Custom endpoints for writer and reader DB instances.
- Same security/monitoring/maintenance features as RDS.
- Aurora Serverless - unpredictable/intermittent workloads, no capacity planning.
- Aurora Global - up to 16 DB Read Replicas in each region, < 1s storage replication.
- Aurora ML - perform ML using SageMaker & Comprehend on Aurora.
- Aurora Database Cloning - new cluster from existing one, faster than restoring a snapshot.
- Used for same as RDS, but with less maintenance/more flexibility/more performance/more features.

▼ ☑ **ElastiCache Summary.**

- Managed Redis/Memcached (similar offering as RDS, but for caches).
- In-memory data store, sub-millisecond latency.
- Select an ElastiCache instance type (e.g, cache.m6g.large).
- Support for Clustering (Redis) and Multi AZ, Read Replicas (sharding).
- Security through IAM, Security Groups, KMS, Redis Auth.
- Backup/Snapshot/PITR feature.
- Managed and scheduled maintenance.
- **Requires some application code changes to be leveraged**.
- Used for key/value store, frequent reads, less writes, cache results for DB queries, store session data for websites, cannot use SQL.

▼ ☑ **DynamoDB Summary.**

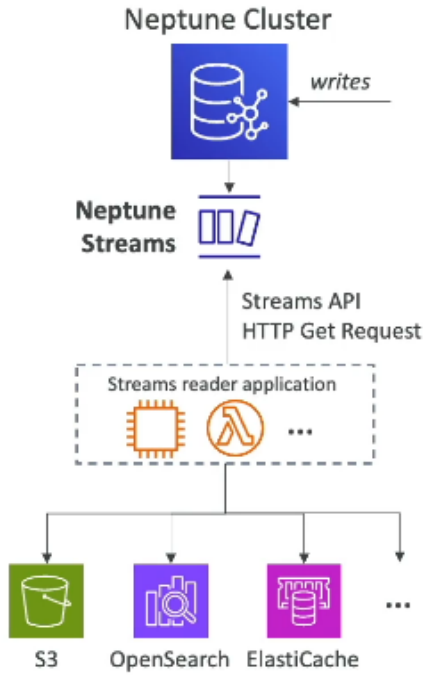
- AWS proprietary technology, managed serverless NoSQL database, millisecond latency.
- Capacity modes: provisioned capacity with optional auto scaling or on-demand capacity.
- Can replace ElastiCache as a key/value store (storing session data for example, using TTL).
- Highly available, Multi-AZ by default, read & writes are decoupled, transaction capability.
- DAX cluster for read cache, microsecond read latency.
- Security, authentication and authorization is done through IAM.
- Event processing: DynamoDB Streams to integrate with AWS Lambda or KDS.
- Global Table feature: active-active setup.
- Automate backups up to 36 days with PITR or on-demand backups.
- Export to S3 without using RCU within the PITR window, import from S3 without using WCU.
- Great to rapidly evolve schemas.
- Used for serverless applications development and distributed serverless cache.

▼ ☑ **Amazon Neptune.**

Neptune is a fully managed graph database. It is highly available across 3 AZ, with up to 15 Read Replicas. We can build and run apps working with highly connected datasets (optimized for these complex and hard queries). It can store up to billions of relations and query the graph with milliseconds latency.

Neptune Streams - real-time ordered sequence of every change to our graph data. Changes are available immediately after writing. There are no duplicates and order is strict. Streams data is accesible in an HTTP REST API.

Use cases: send notifications when certain changes are made, maintain our graph data synchronized in another data store, replicate data across regions in Neptune.



▼ ☑ **Amazon Keyspaces.**

Keyspaces is a managed Apache Cassandra-compatible database service. It is serverless, scalable, highly available, fully managed by AWS. Tables are replicated 3x across multiple AZ.

It uses CQL (Cassandra Query Language) and has single-digit millisecond latency at any scale. For capacity we have on-demand mode or provisioned mode with auto-scaling.

Use cases: store IoT devices info, time-series data, etc...

▼ ☑ **Amazon DocumentDB.**

DocumentDB is "AWS implementation" of MongoDB. It has similar deployment concepts as Aurora (fully managed, highly available with replication across 3 AZ). Its storage automatically grows in increments of 10GB.

Automatically scales to workloads with millions of requests per second.

▼ ☑ **Amazon QLDB & Managed Blockchain.**

QLDB (Quantum Ledger Database) is a fully managed, serverless, high available (across 3 AZ) ledger database that provides a transparent, **immutable** (no entry

can be removed or modified), and cryptographically verifiable transaction log. In QLDB there is no decentralization.

It's used to review history of all the changes made to our application data over time.

Has 2-3x better performance than common ledger blockchain frameworks.

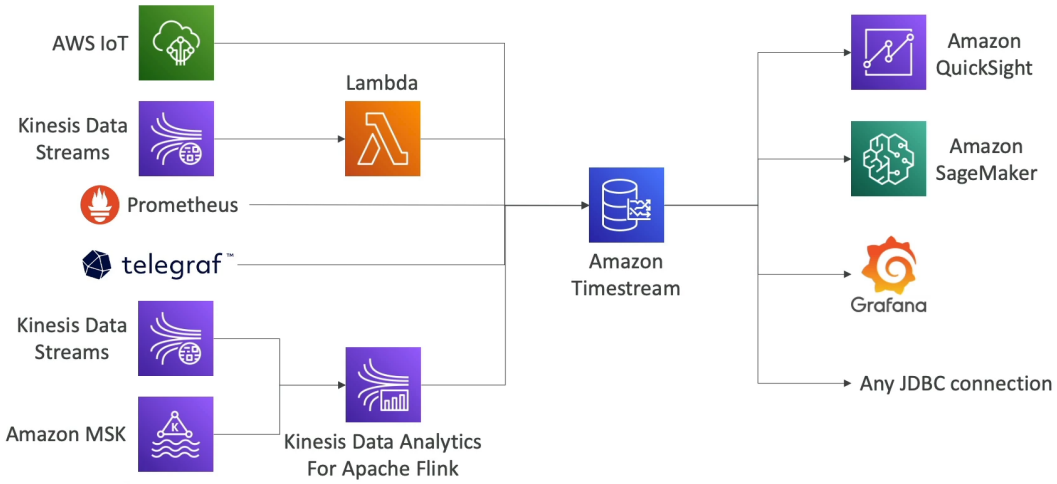
Managed Blockchain is a fully managed decentralized service that simplifies the creation and management of scalable blockchain networks. It is compatibile with Hyperledger Fabric & Ethereum.

▼ ☑ **Amazon Timestream.**

Timestream is a fully managed, fast, scalable serverless time series database. It autimatically scales up/down to adjust capacity. Can store and analyze trillions of events per day. It is 1000s times fasters & 1/10th the cost of relational databases.

It is compatible with SQL and we can use scheduled queries. Recent data is kept in memory and historical data kept in a cost-optimized storage. It has encryption in transit and at rest.

Use cases: IoT apps, operational apps, real-time analytics, etc...



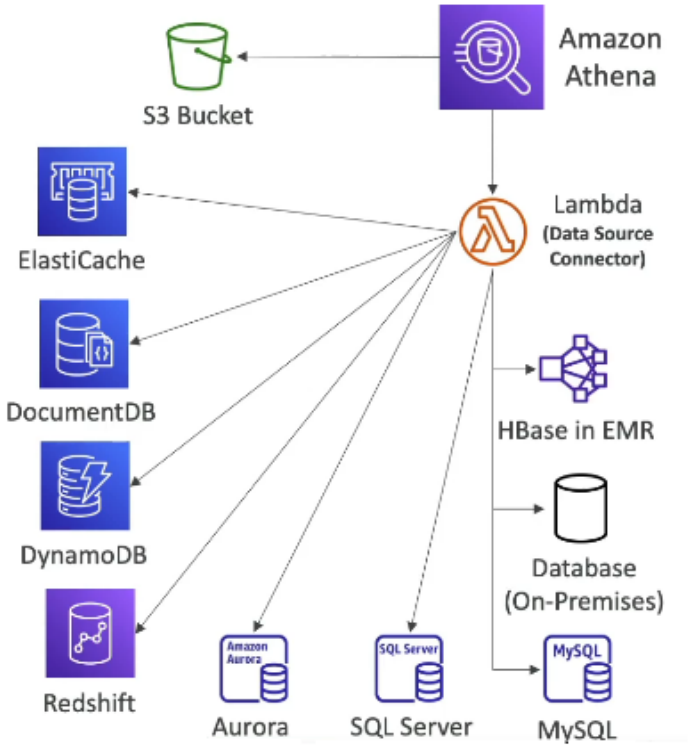
Data & Analytics

▼ ☑ **Amazon Athena.**

Athena is a serverless query service which is used to analyze data stored in S3. It uses standard SQL to query the files and supports CSV, JSON, ORC, Avro and Parquet. It is commonly used with Quicksight for reporting/dashboards.

Use cases: business intelligence, analytics, reporting, analyzing querying VPC Flow Logs, CloudTrail trails, etc...

Athena Federated Query - allows us to run SQL queries across data stored in relational, non-relational, object and custom data sources. It uses Data Source Connectors that run on AWS Lambda and it stores results back in S3.



Athena Performance Improvement

- Use columnar data for cost-savings (less scan) - Apache Parquet or ORC is recommended. We can use Glue to convert our data to Parquet or ORC.

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	130 GB	6.78 seconds	2.51 GB	$0.01
Savings / Speedup	87% less with Parquet	34x faster	99% less data scanned	99.7% savings

- Compress data - for smaller retrievals.
- Partition datasets in S3 - easy querying on virtual columns (*s3://athena-ex/flight/parquet/year=1995/month=1/day=1*).
- Use larger files (> 128MB) to minimize overhead.

▼ ☑ **Amazon Redshift.**

Redshift is based on Postgres, but it's not used for OLTP (Online Transaction Processing). It is used for **OLAP** (Online Analytical Processing - Analytics & Data

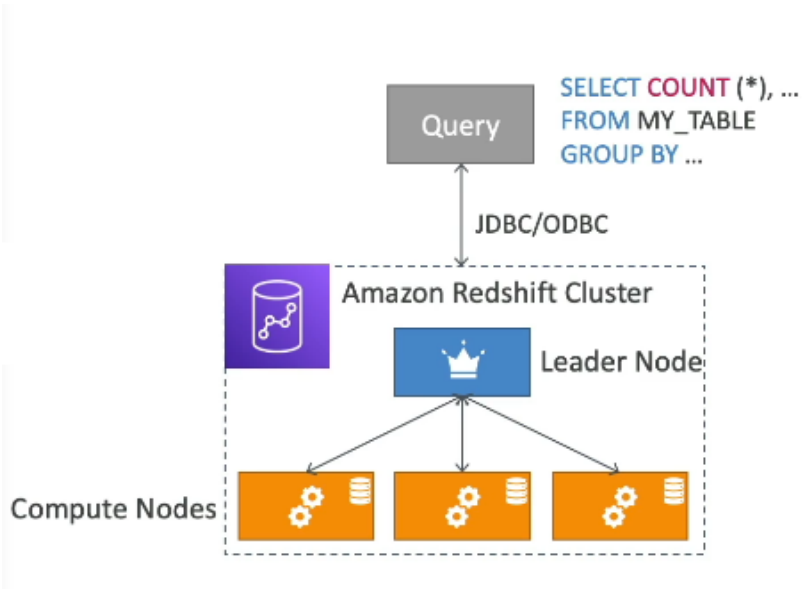
Warehousing). Redshift loads data once every hour, not every second and it has 10x better performance than other data warehouses, scale to PBs of data. Has Massively Parallel Processing (MPP), highly available.

Instead of row based storage of data Redshift uses columns. In comparison with Athena, Redshift has faster queries, joins and aggregations because it uses indexes.

Redshift has pay as you go type of pricing based on the instances provisioned. By leveraging **cross-region snapshot copying**, we can significantly enhance our Redshift cluster's resilience and ensure that our data is protected against regional disruptions.

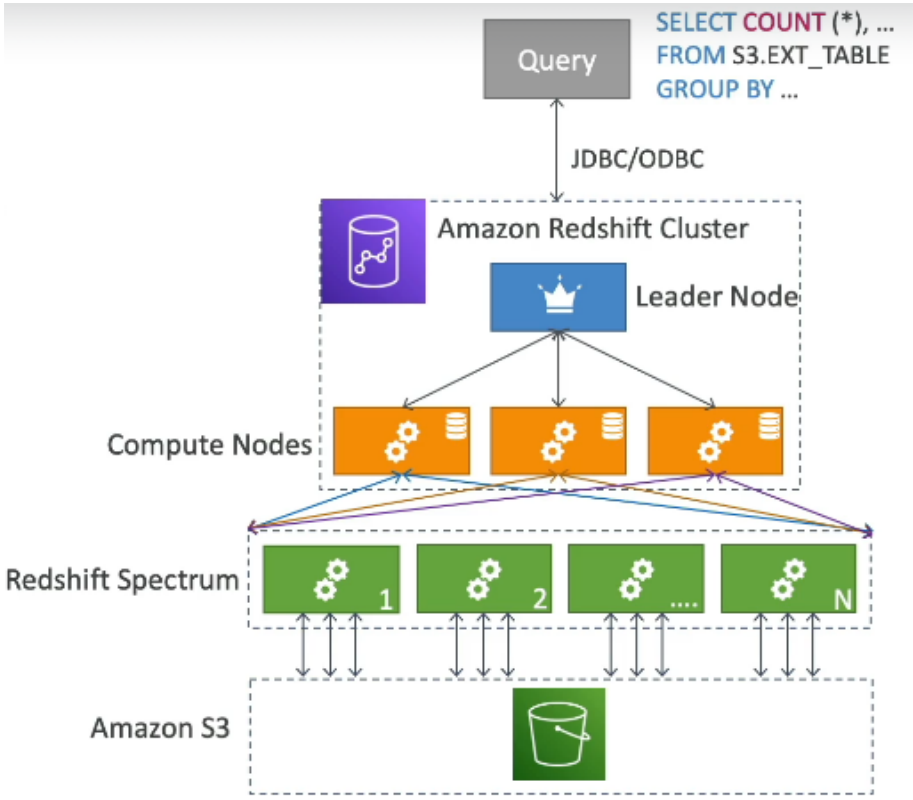
Redshift Cluster

- **Leader Node** - for query planning and results aggregation.
- **Compute Node** - for performing the queries and sending results to leader.



We must provision the node size in advance and use RI for cost savings.

Redshift Spectrum - feature which lets us query data that is already in S3 without loading it. For this we must have a Redshift cluster available to start the query.

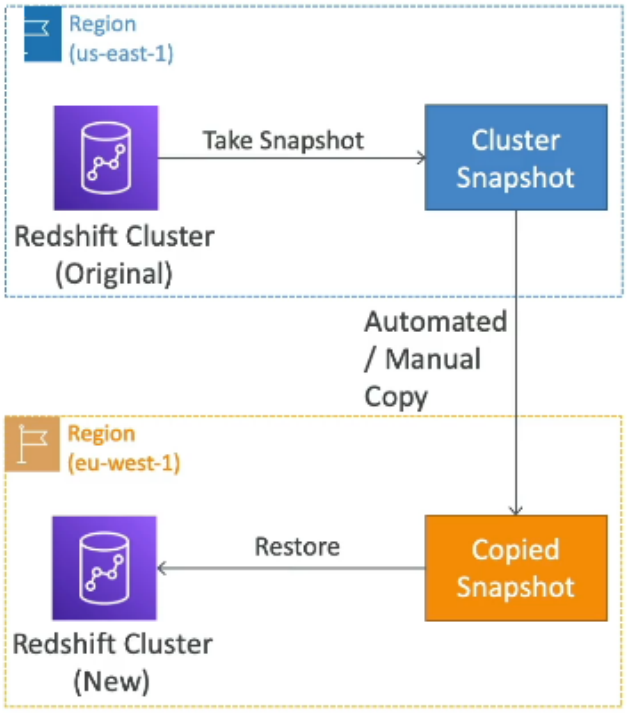


Redshift Snapshots & DR

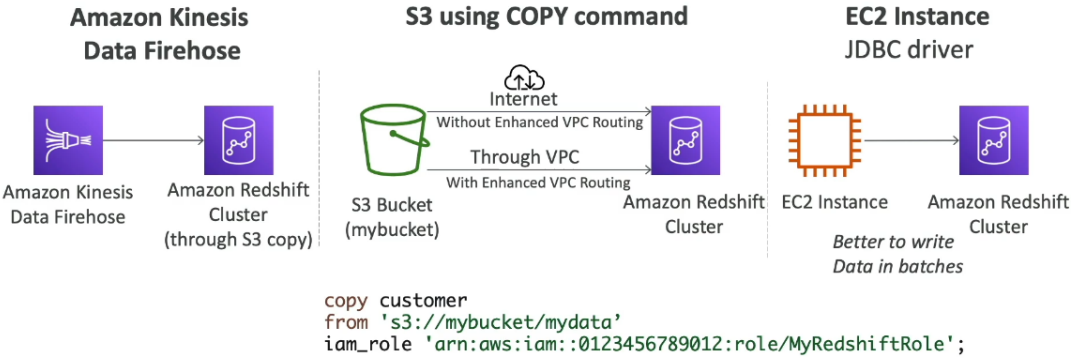
Redshift has Multi-AZ mode for some clusters. Snapshots are PIT backups of a cluster, stored internally in S3. They are incremental (only what has changed is saved). We can restore a snapshot into a new cluster.

- **Automated snapshots** - every 8h, every 5GB or on a schedule. Need to set retention period.
- **Manual snapshots** - snapshot is retained until we delete it.

We can configure Redshift to automatically copy snapshots of a cluster to another AWS Region.



Loading Data into Redshift



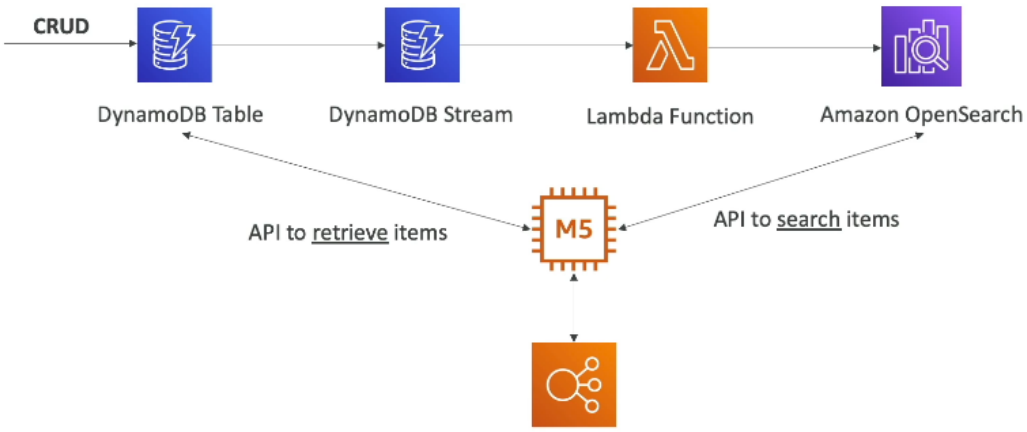

```
copy customer
from 's3://mybucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

▼ ☑ Amazon OpenSearch (ElasticSearch).

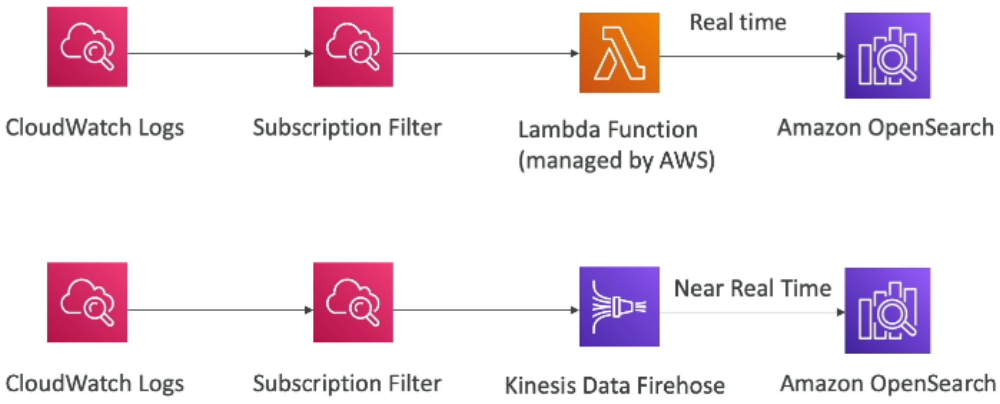
With **OpenSearch** we can search any field, even partially matches. It's common to use OpenSearch as a complement to another database. It does not natively support SQL (can be enabled via a plugin). It comes with OpenSearch Dashboards for visualization.

There are two modes: managed cluster and serverless cluster.

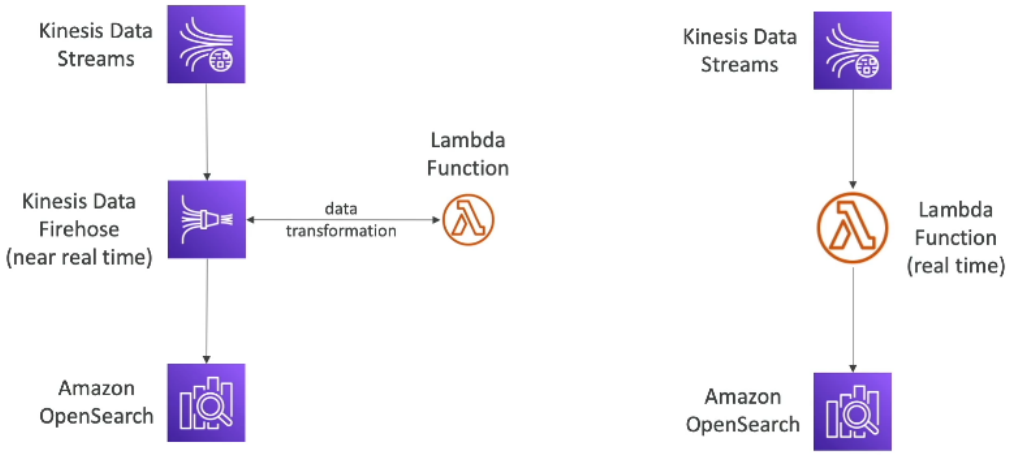
▼ **Solution Architecture - OpenSearch DynamoDB Integration.**



▼ **Solution Architecture - OpenSearch ClodWatch Logs Integration.**



▼ **Solution Architecture - OpenSearch KDS & KDF Integration.**



▼ ☑ **Amazon EMR.**

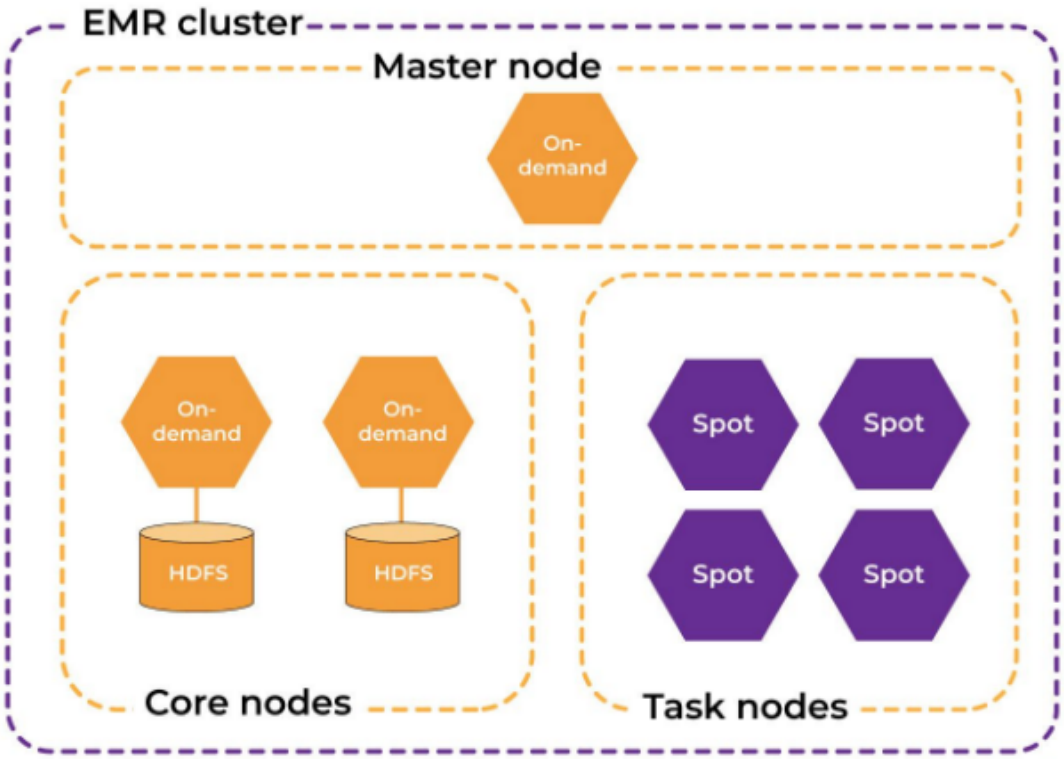
EMR helps creating Hadoop clusters (Big Data) to analyze and process vast amount of data. The clusters can be made of hundreds of EC2 instaces. It takes care of all the provisioning and configuration. It has auto-scaling and it's integrated with Spot instances.

EMR supports Apache Spark, HBase, Presto, Flink...

Usage: data processing, log analysis, ML, web indexing, big data ...

EMR Node Types

- **Master Node** - manages and coordinates the cluster (long running).
- **Core Node** - runs tasks and stores data (long running).
- **Task Node** (optional) - just to run tasks (short running).



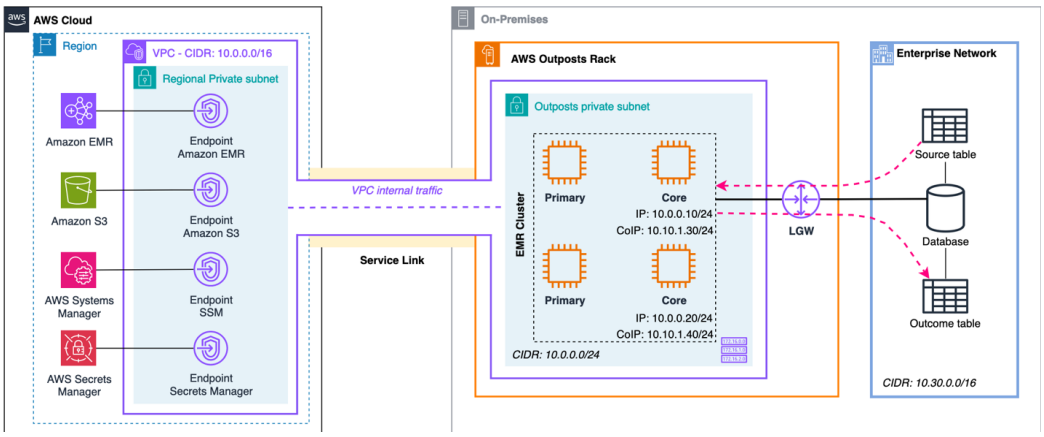
EMR Cluster Types

- **Long-running Cluster** - designed for ongoing data processing needs. They remain active for extended periods, allowing users to submit multiple jobs over time without the overhead of cluster startup and configuration.
- **Transient Cluster** - designed for short-lived, one-off jobs. They are created for specific processing tasks and terminated immediately after the job is complete.

Purchasing Options

- **On-demand** - reliable, predictable, wont be terminated.
- **Reserved** (min 1 year) - cost savings (EMR will automatically use if available).
- **Spot Instances** - cheaper, can be terminated, less reliable.

▼ **Solution Architecture - Process Data from an On-Premises Database.**

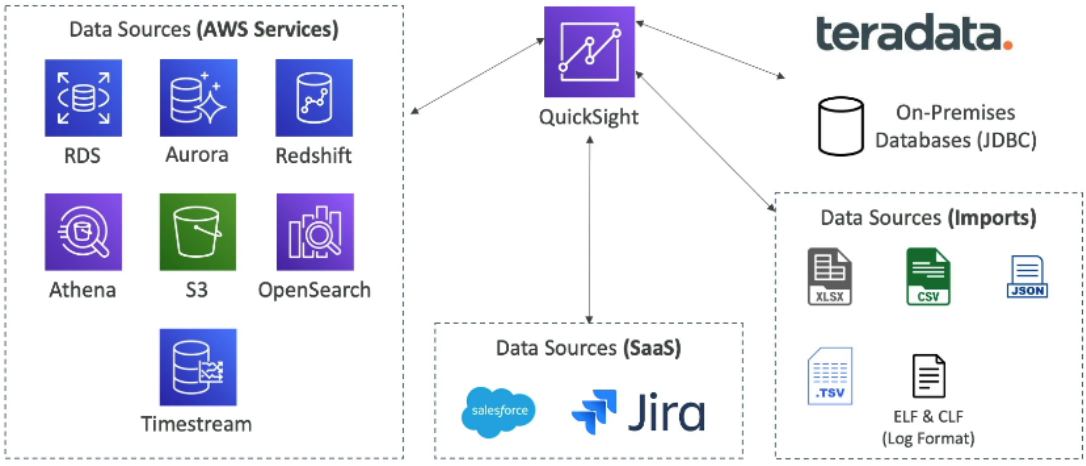


▼ ☑ **Amazon QuickSight.**

QuickSight is a serverless ML-powered business intelligence service to create interactive dashboards. It is fast, automatically scalable, embeddable, with per-session pricing. Can be integrated with RDS, Aurora, Athena, Redshift ...

It has in-memory computation capabilities using SPICE engine. In enterprise edition we have possibility to setup Column-Level Security (CLS) to prevent some columns to be displayed to users who dont have permission.

Usage: business analytics, building visualizations, perform ad-hoc analysis ...



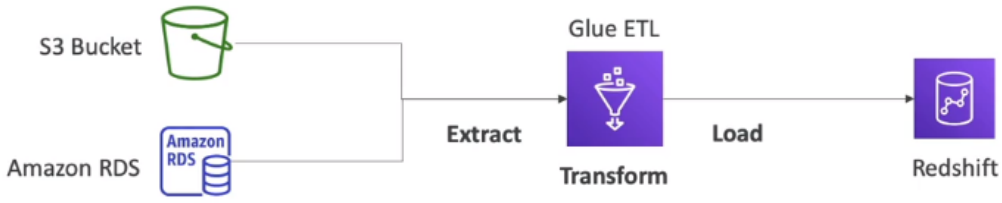
In QuickSight we can define users (standard versions) and groups (enterprise version). These users and groups only exist within QuickSight, not IAM!

Dashboard - read-only snapshot of an analysis that we can share. It preserves the configuration of the analysis.

We can share the analysis or the dashboard with users or groups. Users who see the dashboard can also see the underlying data.

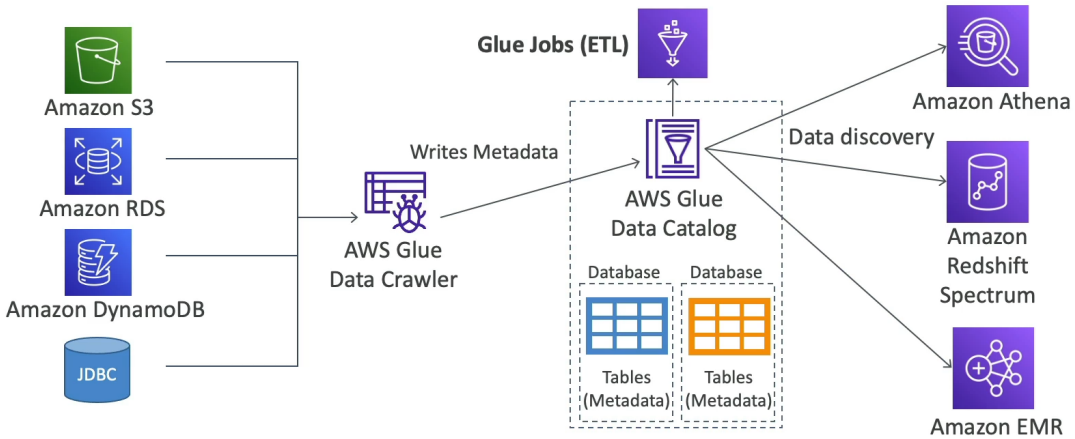
▼ ☐ **AWS Glue.**

AWS Glue is a serverless fully managed extract, transform, and load (**ETL**) service. It is useful for preparing and transforming data for analytics.



Glue Components

- **Data Catalog** - central repository that stores metadata about datasets in our AWS environment. Can be used by Athena, Redshift or EMR.
- **ETL Engine** - automatically generates Python or Scala code to extract data from various sources, transform it, and load it into target data stores.
- **Crawlers** - automated processes in AWS Glue that scan and infer the schema of data stored in various sources.
- **Glue Jobs** - ETL scripts that users create to transform data. These jobs can be scheduled and monitored within AWS Glue.



Glue Advanced Features

Glue Job Bookmarks - save and track the data that has already been processed, prevent re-processing old data.

Glue Elastic Views - combine and replicate data across multiple data stores using SQL. No custom code is needed, Glue monitors for changes in the source data (serverless).

Glue DataBrew - clean and normalize data using pre-built transformation.

Glue Studio - used to create, run and monitor ETL jobs in Glue.

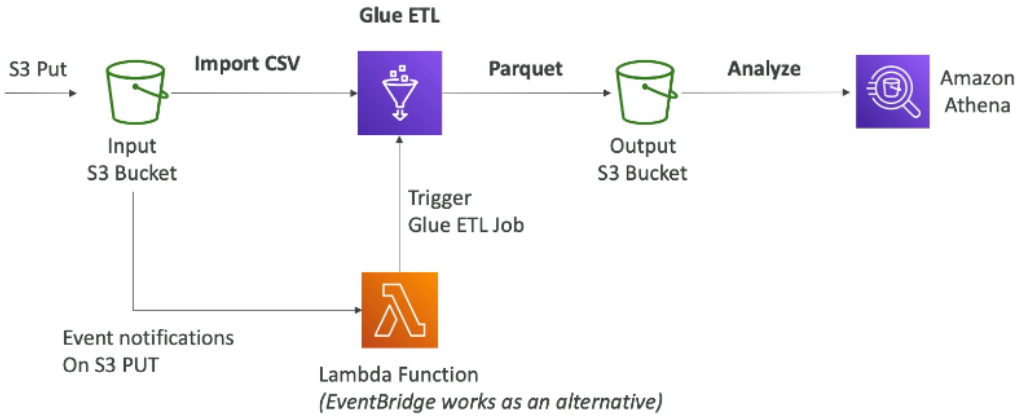
Glue Streaming ETL - run Glue jobs as streaming jobs. compatible with KDS, Kafka, MSK

Glue PySpark Jobs

PySpark provides distributed processing via Apache Spark, which makes Glue ideal for handling large datasets and complex transformations.

Feature	AWS Glue PySpark Jobs	AWS Lambda
Data Size	Large-scale data (GBs-TBs)	Small-medium data (up to a few GBs, constrained by memory and timeout limits)
Execution Time	Long-running jobs (up to hours)	Short-lived tasks (maximum 15 minutes)
Processing Type	Batch processing, complex ETL	Real-time, event-driven processing
Memory	Large memory and distributed computing available via Spark	Limited (up to 10 GB of memory)
Scalability	Automatically scales for large, distributed workloads	Automatically scales but limited by function constraints
Cost	Typically cost-effective for large datasets and long-running jobs	Cost-effective for short-lived, low-latency tasks
Integration with Other Services	Tight integration with S3, Redshift, RDS, Athena	Integration with many AWS services, especially for event-driven triggers

▼ **Solution Architecture - Convert data into Parquet format.**

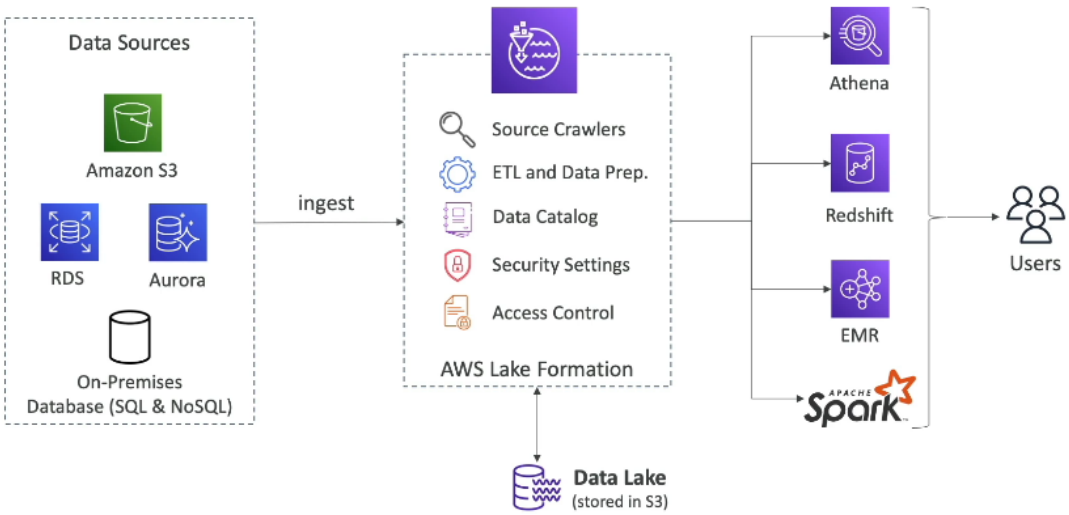


▼ ☑ **AWS Lake Formation.**

Lake Formation is a fully managed service that makes it easy to setup a data lakes in days. Data Lake is a central place to have all our data for analytics purposes. It is built on top of AWS Glue.

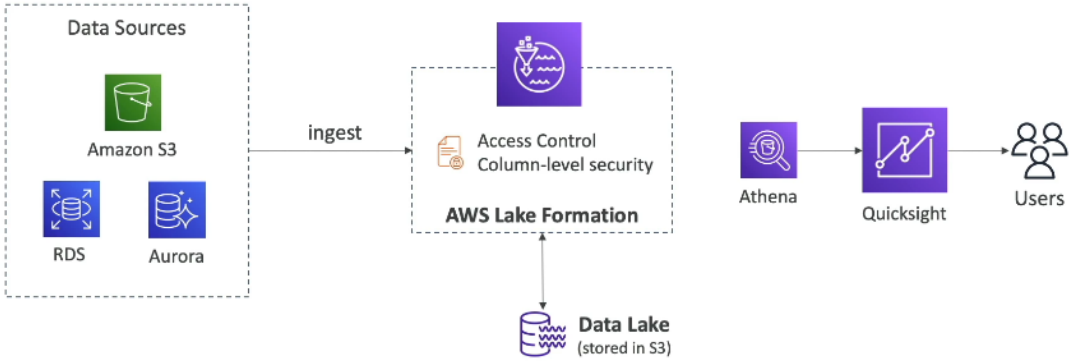
With Lake Formation we can discover, cleanse, transform and ingest data. It combines structured and unstructured data. It has blueprints for S3, RDS, relational & NoSQL DB, etc. Blueprints can help us migrate data from one place to the data lake.

We can have fine-grained access controls for our applications at the row and column level.



[Lake Formation - Centralized Permissions]

Lake Formation solves problem of security management by centralizing it. AWS Lake Formation simplifies the security of our data lake, making it easier to enforce data governance policies. We can consolidate data from multiple accounts into a single account.

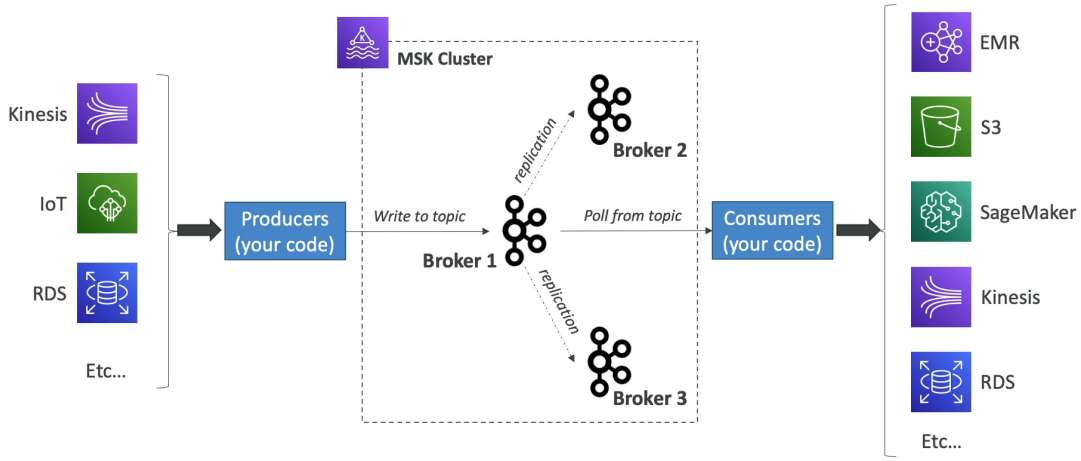


▼ ☑ MSK (Managed Streaming for Apache Kafka).

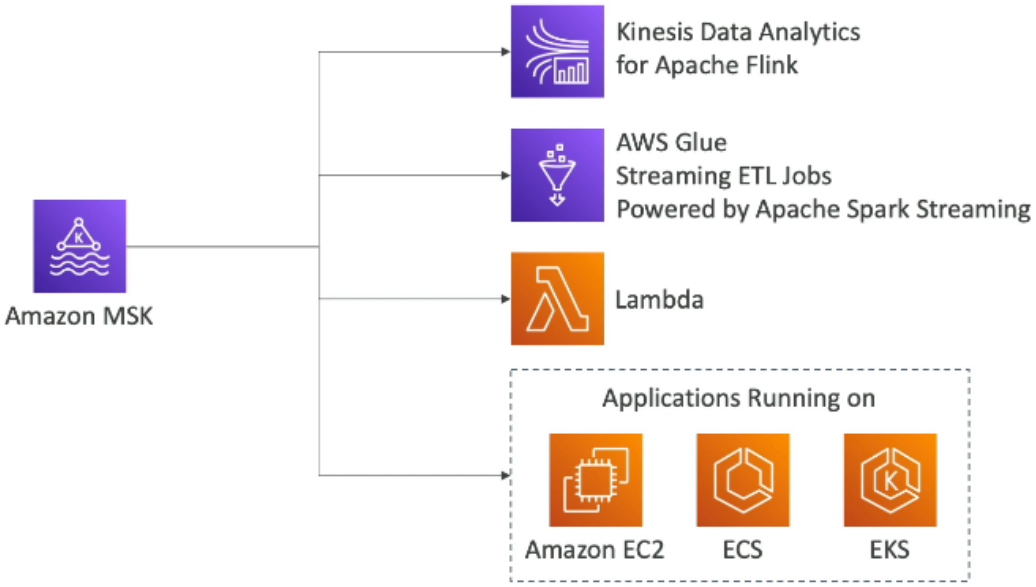
MSK is an alternative to Amazon Kinesis. It is fully managed Apache Kafka on AWS. It allows us to create, update and delete clusters.

MSK creates and manages Kafka brokers nodes and Zookeeper nodes for us. We can deploy the MSK cluster in our VPC and multi-AZ. Data is stored in **EBS** volumes for as long as we want.

MSK Serverless - run Apache Kafka on MSK without managing the capacity.



MSK Consumers

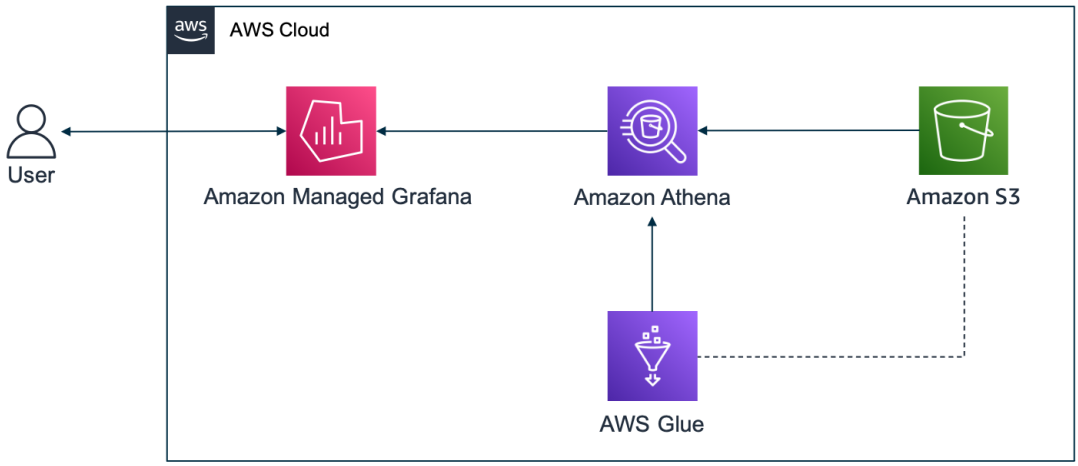


Kinesis Data Streams vs MSK

Feature	Amazon MSK (Managed Streaming for Apache Kafka)	Amazon Kinesis Streams
Technology	Apache Kafka	Kinesis-specific technology
Data Storage	Persistent storage with configurable retention	Configurable retention from 24 hours to 365 days
Sharding	Uses Kafka partitions for scaling	Uses shards for scaling
Message Ordering	Maintains order within a partition	Maintains order within a shard
Integration	Works with Kafka ecosystem tools and connectors	Integrated with AWS services like Lambda, S3
Operational Complexity	Requires understanding of Kafka operations	Simplified operation with AWS management
Cost	Based on broker instance types and storage	Based on data ingestion and retrieval

▼ ☑ **AWS Managed Grafana.**

AWS Managed Grafana is a fully managed service that makes it easy to deploy, operate, and scale Grafana for monitoring and observability. Grafana is a popular open-source analytics and monitoring tool that integrates with a wide variety of data sources to provide customizable dashboards and visualizations for metrics, logs, and traces. It scales automatically based on our needs.



AWS Managed Grafana integrates with IAM and supports SSO via AWS SSO and other identity providers. It is ideal for organizations that need to monitor a variety of data sources across different environments, including on-premises, cloud-native, and hybrid environments.

Machine Learning

▼ ☑ **ML AWS Services.**

- **Rekognition**

  Finds objects, people, text, secenes in images and videos using ML. Can be used for facial anaylsis and facial search to do user verification, people counting ...

- **Transcribe**

  Automatically converts speech to text using automatic speech recognition (ACR). It can automatically remove personally identifiable information using Redaction. Supports multi-lingual audio.

- **Polly**

  Turns text into speech using deep learning. It allows us to create applications that talk.

- **Translate**

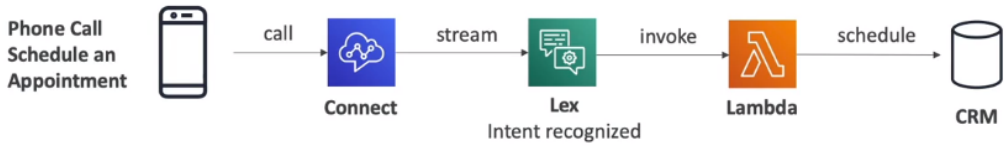
It provides us natural and accurate language translation. Allows us to **localize content** - such as websites and applications for international users.

- **Lex**

  Same technology that powers Alexa. It's used for ASR and NLP, helps in building chatbots and call center bots.

- **Connect**

  Used to receive calls, create contact flows, cloud-based virtual contact center. No upfront payments, 80% cheapter than traditional contact center solutions.

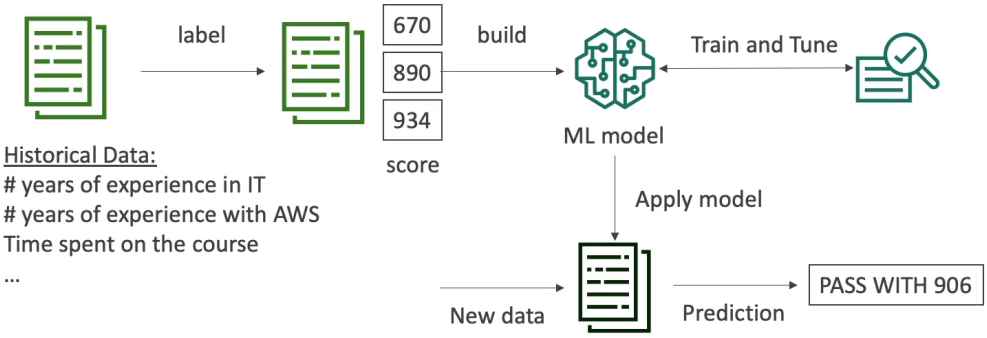


- **Comprehend**

  Used for NLP, fully managed and serverless service. Used for extracting key phrases, places, people ... Analyzes text using tokenization and parts of speech, andautomatically organizes a collection of text files by topic.

- **SageMaker**

  Fully managed service for developers and data scientists to build ML models.

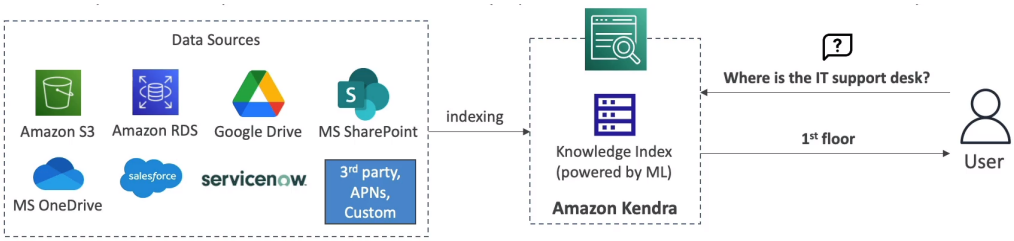


- **Forecast**

Fully managed service that uses ML to deliver highly accurate forecasts. It's 50% more accurate than looking at the data itself.

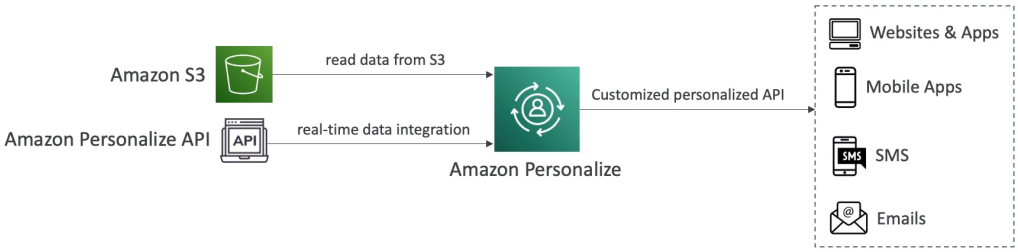
- **Kendra**

  Fully managed document search service powered by ML. It extracts answers from within a documents (text, pdf, HTML ...). It can learn from user interactions/feedback to promote preferred results (Incremental Learning).



- **Presonalize**

  Fully managed ML service to build apps with real-time personalized recommendations. Can be integrated into existing websites and applications.



- **Textract**

  Automatically extracts text, handwriting and data from any scanned documents using ML.

- **Monitron**

Detects abnormal conditions in industrial machinery, enabling us to implement predictive maintenance and reduce unplanned downtime.

- **Panorama**

  Aims to bring computer vision (CV) capabilities to the edge devices in industrial environments.

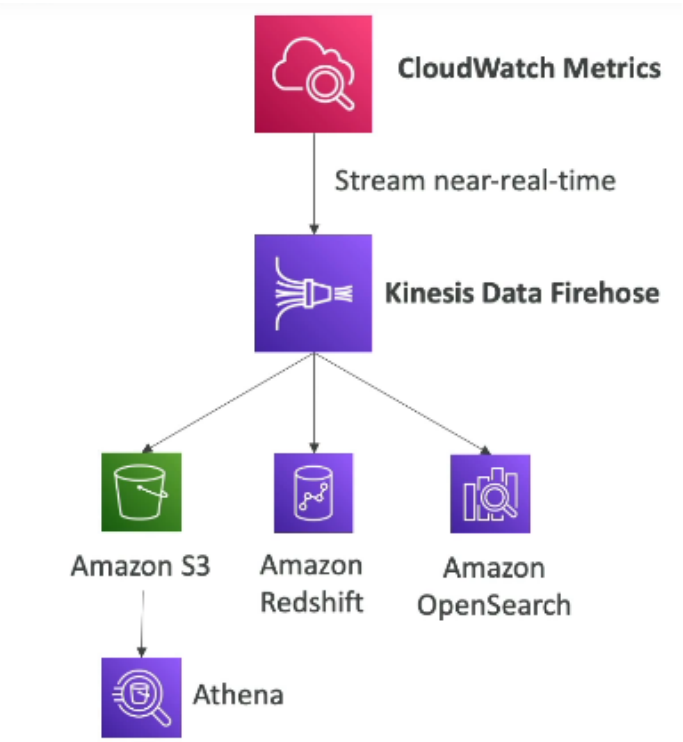
AWS Monitoring & Audit

▼ ☑ **CloudWatch Metrics.**

CloudWatch provides metrics for every service in AWS. Metric is a variable to monitor (CPU Utilization, Network In, etc). We can create custom metrics too (**Memory Utilization**).

Each metric has timestamps and dimension. **Dimension** is an attribute of a metric (instance id, environment, etc.). CloudWatch can create dashboards of metrics.

CloudWatch Metric Streams - continually stream CloudWatch metrics to a destination of our choice, with near real-time delivery and low latency. We can use KDF or third party service provider (Datadog, Dynatrace, etc). We have option to filter metrics to only stream a subset of them.



▼ ☐ **CloudWatch Logs.**

CloudWatch Logs is a logging service that monitors, stores, and access log files from AWS resources, applications, and services. Logs are encrypted by default.

- Log groups - arbitrary name, usually representing an application.
- Log stream - instance within applications/log files/containers.

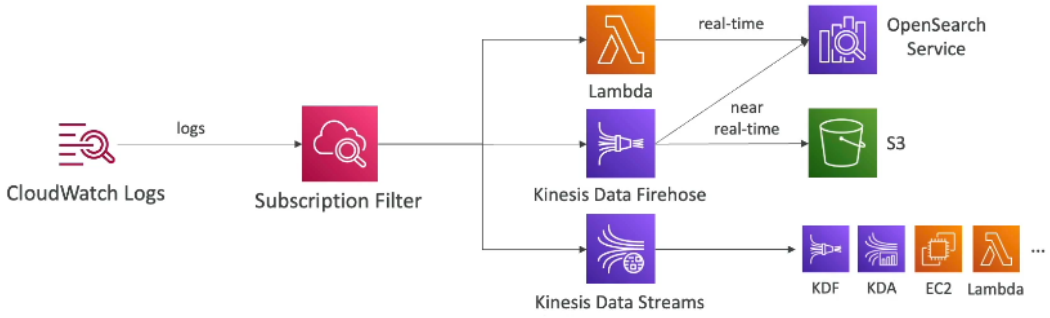
CloudWatch Logs can send logs to: S3, KDS, KDF, AWS Lambda, OpenSearch.

Logs can be collected from: Elastic Beanstalk, ECS, AWS Lambda, CloudTrail, CloudWatch Log agents (on EC2 instances or on-premises servers), Route53.

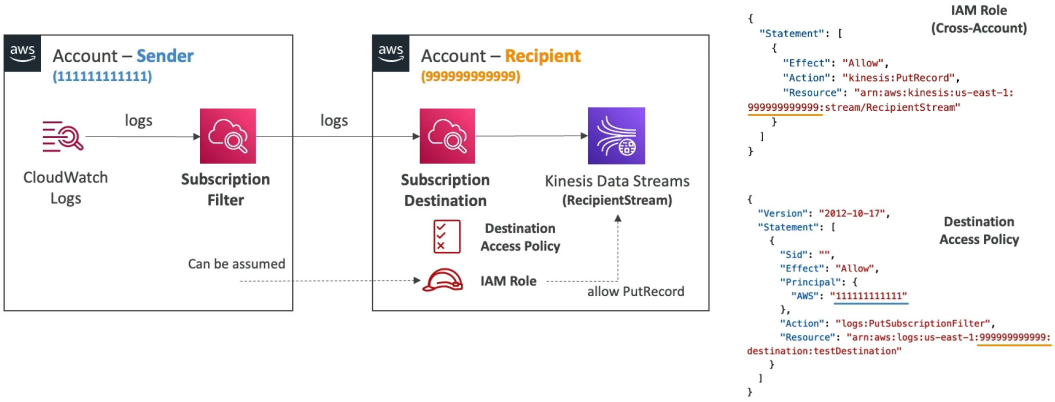
CloudWatch Logs Insights - search and analyze log data stored in CloudWatch Logs. Provides a purpose-built query language. It automatically discovers fields from AWS services and JSON log events. We can save queries and add them to CloudWatch Dashboard. It's query engine, not a real-time engine.

Log data can take up to 12h to become available for export. It is not near real-time or real-time, if we want that we should use Logs Subscriptions instead.

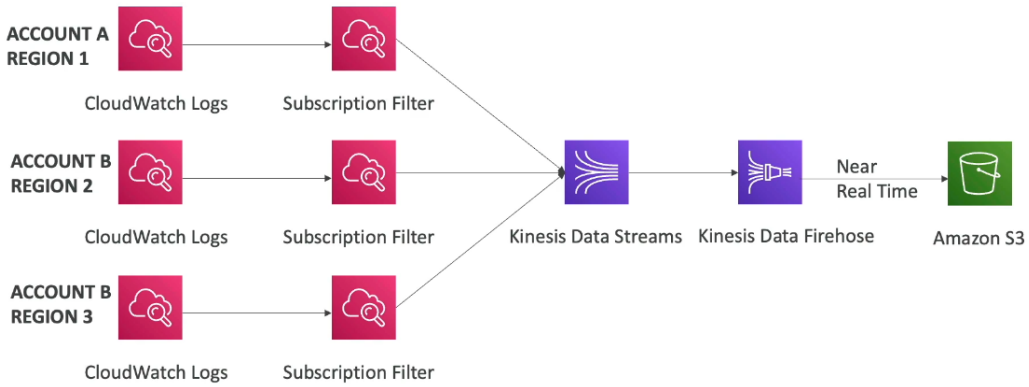
CloudWatch Logs Subscriptions - lets us get a real-time log events from CloudWatch Logs for processing and analysis. We can send to KDS, KDF or AWS Lambda. There is Subscription Filter which can filter which logs will be delivered to our destination.



We can have Cross-Account Subscription which can send log events to resources in different AWS account (KDS, KDF).

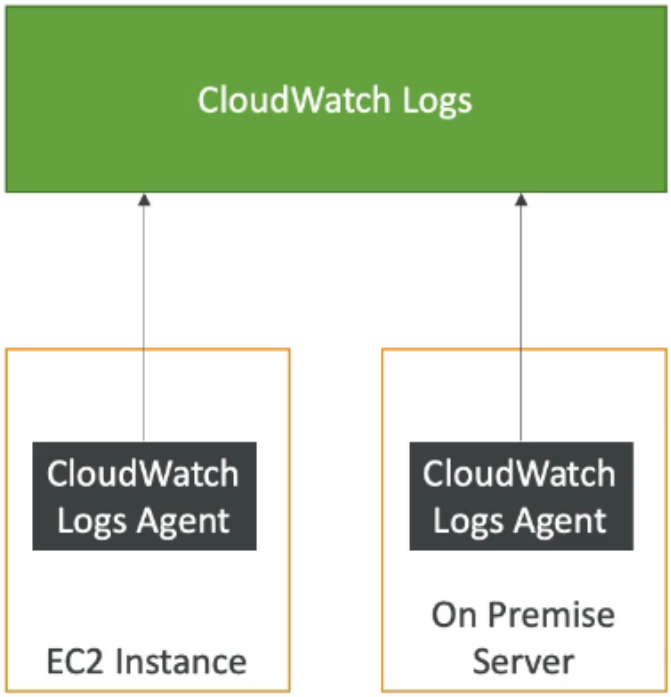


▼ **Solution Architecture - Logs Aggregation (Multi-Account & Multi Region).**



▼ ☑ **CloudWatch Agent & CloudWatch Logs Agent.**

By default, no logs from our EC2 instance will go to CloudWatch, we need to run a CloudWatch Agent on EC2 to push the log files we want (need to make sure IAM permissions are correct).



CloudWatch Logs Agent - old version of the agent. Can only send to CloudWatch Logs.

CloudWatch Unified Agent - it collects additional system-level metrics such as RAM, processes, etc. It collect logs to send them to CloudWatch Logs. It has

centralized configuration using SSM Parameter Store.

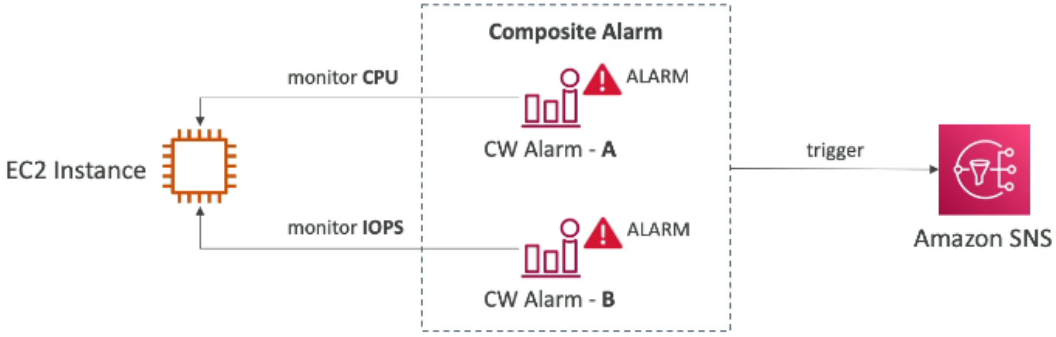
▼ ☐ **CloudWatch Alarms.**

CloudWatch Alarms - used to trigger notifications for any metrics. It has various options (sampling, percentage, max, min, etc). With CloudWatch Alarm we can stop, terminate, reboot or recover an EC2 instance, trigger auto scaling action or send notification to SNS.

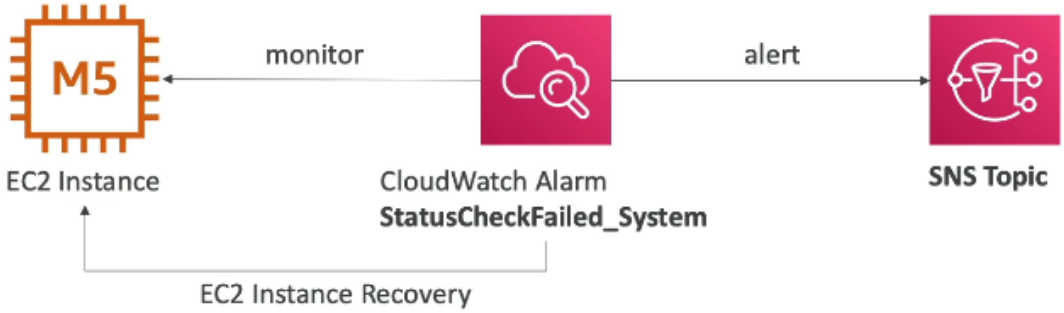
Alarm States: OK, INSUFFICIENT_DATA, ALARM.

Period - length of time in seconds to evaluate the metrics.

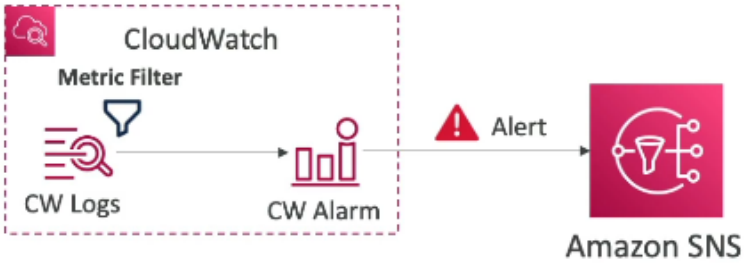
Composite Alarms - monitoring the states of multiple other alarms and we can use AND and OR conditions to merge them.



There is a status check to check the EC2 and we can define a CloudWatch Alarm to monitor a specific EC2 instance. In case the alarm is being breached, we can start an EC2 instance recovery or send an alert to our SNS topic.



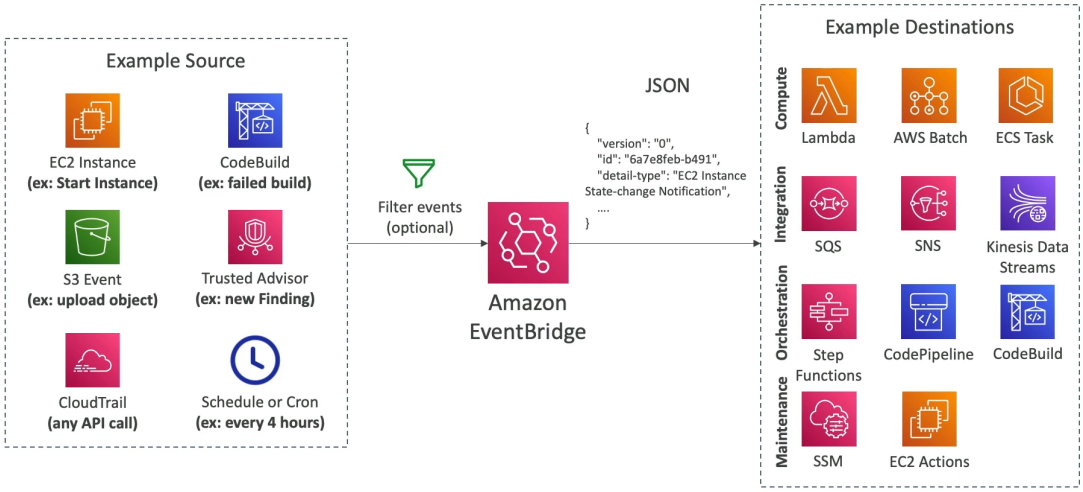
Alarms can be created based on CloudWatch Logs Metrics Filters.



▼ ☐ **CloudWatch Events (EventBridge).**

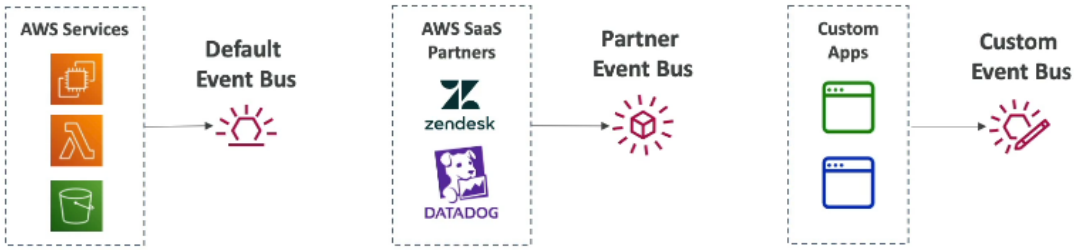
EventBridge is a serverless event bus service that makes it easy to connect applications using data from our own applications, integrated SaaS applications, and AWS services.

We can schedule events with CRON jobs or we can make event rules to react to a service doing something. Also it's possible to trigger Lambda functions, send SQS/SNS messages ...



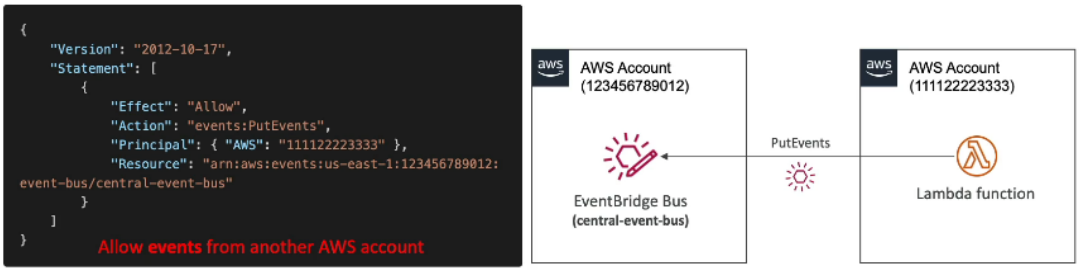
EventBridge Schema Registry - allows us to generate code for our application, that will know in advance how data is structured in the event bus. Schema can be versioned.

Event Bus Types



Event buses can be accessed by other AWS accounts using resource-based policies.

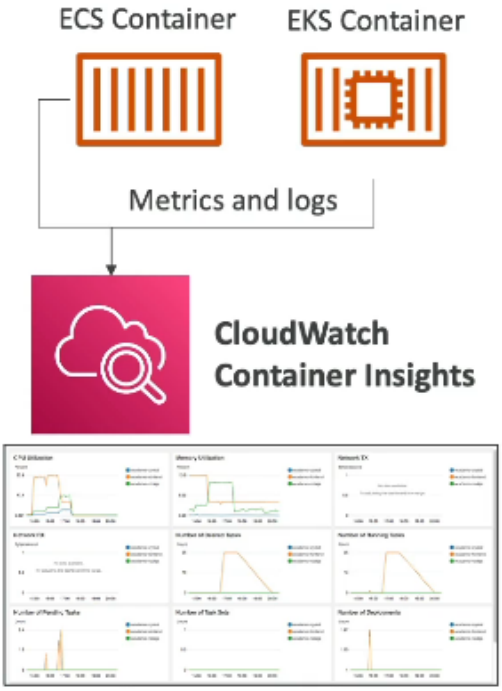
Resource-based Policy - manages permissions for a specific Event Bus. It is used to aggregate all events from our AWS Organization in a single AWS account or AWS region.



We can archive events sent to an event bus (indefinetely or set period) and we can replay archived events.

▼ ☐ **CloudWatch Insights & Operational Visibility.**

CloudWatch Container Insights - used to collect, aggregate and summarize metrics and logs from containers. In EKS and Kubernetes, CloudWatch Insights is using a containered version of the CloudWatch Agent to discover containers. We cant use it on-premises.



CloudWatch Lambda Insights - monitoring and troubleshooting solution for serverless applications running on AWS Lambda. It collects, aggregates and summarizes system-level metrics and diagnostic information (cold starts, Lambda worker shutdowns).

CloudWatch Contributor Insights - analyzes log data and creates time series that display contributor data. We can see metrics about the top-N contributors. It helps us to find top talkers and understand who or what is impacting system performance. We can build our rules from scratch or we can also use sample rules that AWS has created.

Use cases: find bad hosts, identify the heaviest network users or find the URLs that generate the most errors.

CloudWatch Application Insights - provides automated dashboards that show potential problems with monitored applications to help isolate ongoing issues. It is powered by SageMaker.

It gives us enhanced visibility into our applicaion health to reduce the time it will take us to troubleshoot and repair our applications. Findings and alerts are sent to EventBridge and SSM OpsCenter.

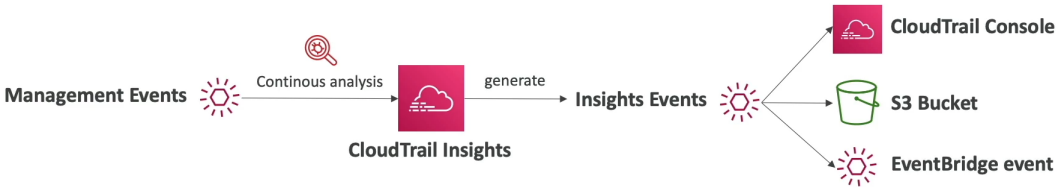
▼ ☑ CloudTrail.

AWS CloudTrail provides governance, compliance and audit for our AWS account. It is enabled by default. We can get history of events and API calls made within our account. It's possible to put logs from CloudTrail into CloudWatch Logs or S3. Can be applied to all regions (default) or a single region. By default, CloudTrail event log files are encrypted using Amazon S3 server-side encryption (SSE).

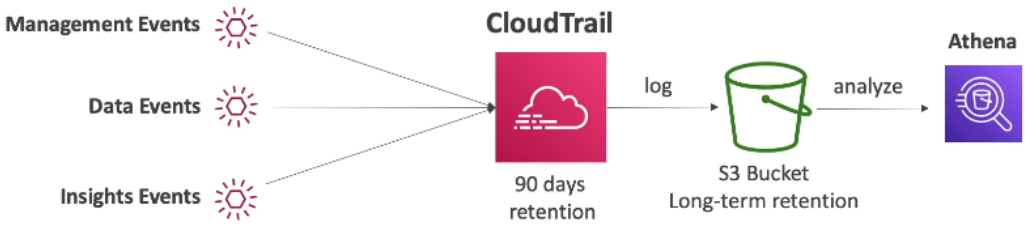
CloudTrail Events

- **Management Events** - operations that are performed on resources in our AWS account. By default, trails are configured to log management events. We can separate read events from write events.
- **Data Events** - S3 object-level activity and Lambda function execution activity (get, delete, put). By default, data events are not logged because high volume operations. It can separate read events from write events.
- **CloudTrail Insights Events**

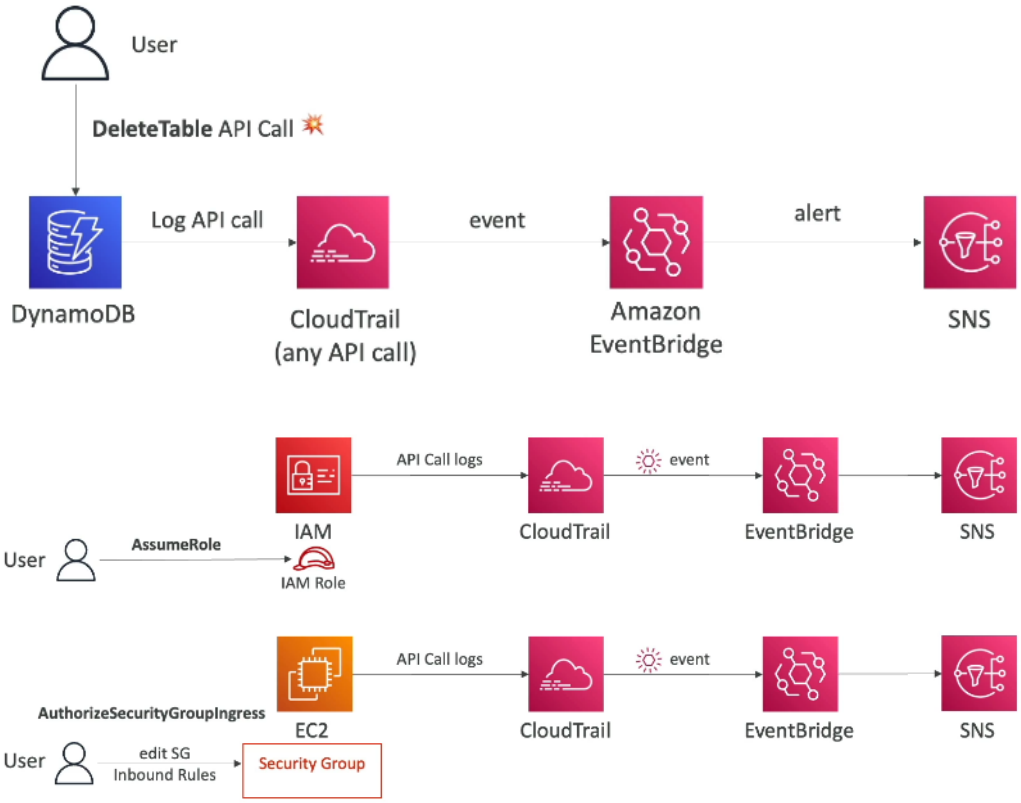
CloudTrail Insights is a feature within AWS CloudTrail that helps us detect unusual or unexpected activity in your AWS accounts. It continuously analyzes our CloudTrail event logs to identify patterns and anomalies, enabling us to quickly respond to potential security issues.



Events are stored for 90 days in CloudTrail. To keep events beyond this period we should log them to S3 and use Athena.



▼ **Solution Architecture - Intercept API Calls (CloudTrail + EventBridge)**



▼ **☑ AWS Config.**

AWS Config helps us with auditing and recording compliance of our AWS resources. It can record configurations and changes over time. It is possible to store the configuration data into S3 (analyzed by Athena).

Questions that can be solved by AWS Config:

- Is there unrestricted SSH access to my security groups?

- Do my buckets have any public access?
- How has my ALB configuration changed over time?

Config is a per-region service and can be aggregated across regions and accounts. We can receive alerts (SNS notifications) for any changes.

Config Rules

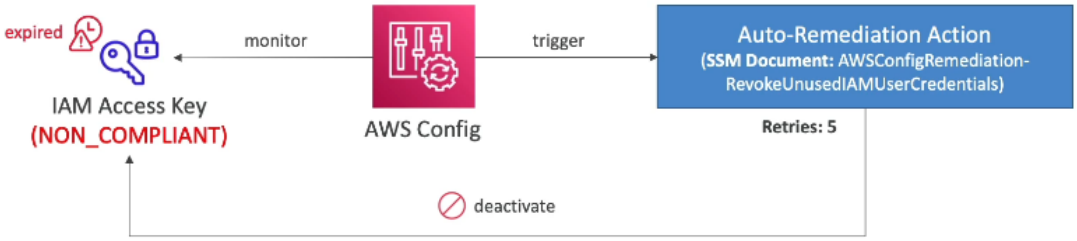
We can use AWS managed config rules or make custom config rules (**must be defined in AWS Lambda**). Rules can be evaulated/triggered for each config change and/or at regular time intervals.

Config Rules does not prevent actions from happening (**no deny**).

Config Remediations

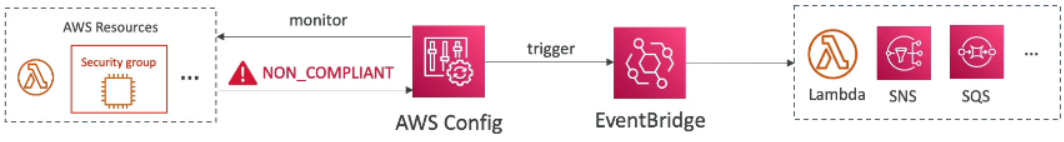
We can automate remediation of non-compliant resource using SSM Automation Documents. It is possible to create custom Automation Documents that invokes Lambda function.

We can set Remediation Retries if the resource is still non-compliant after auto-remediation.



Config Notifications

We can use EventBridge to tigger notifications when AWS resources are non-compliant.



There is ability to send configuration changes and compliance state notifications to SNS.